Universidad Carlos III de Madrid

Departamento de Ingeniería Telemática

PROYECTO FIN DE GRADO

DESARROLLO DE UN GATEWAY DE MAVLINK PARA MICRO DRONES PARA GESTIÓN DE VUELO AUTÓNOMO

Autor: Javier Casado Reina

Tutor: Francisco Valera Pinto

Leganés, septiembre de 2017

Título: Desarrollo de un gateway de MAVLink para micro drones para gestión de vuelo autónomo

Autor: Javier Casado Reina

Director: Francisco Valera Pinto

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Quiero agradecer a mi familia por el gran apoyo que me han dado durante la realización de este proyecto, sobre todo a mis padres y mi hermana que me han estado ayudando en todo lo que pedía.

También necesito dar las gracias a mi tutor, Francisco Valera, por su inestimable ayuda y guía en este proyecto.

Resumen

Desde hace unos años el sector de los drones está teniendo una fuerte evolución, siendo cada vez más las empresas dedicadas a su fabricación. Como consecuencia se ha producido una bajada de precio de los mismos, lo que permite las empresas e incluso el público en general puedan comprarse más de un dron.

En estas circunstancias y debido a que la mayoría de los fabricantes crean sus propios sistemas para comunicarse con sus drones, se presenta para el usuario la dificultad de tener que aprender a utilizar cada uno de ellos de manera independiente.

Para dar solución a este problema, en este proyecto se ha desarrollado una aplicación que, mediante el protocolo de comunicación MAVLink, permite controlar diferentes modelos de drones con una interfaz común para todos ellos.

Por si sola esta funcionalidad es de gran utilidad, pero además, con el fin de lograr una aplicación con mayor potencial comercial, en el proyecto se ha abordado también el desarrollo de dos funcionalidades más: la posibilidad de realizar vuelos simultáneos con más de un dron a la vez, y poder realizar la comunicación con los drones a través de un proxy intermedio para conseguir mayor versatilidad.

Palabras clave: Dron, MAVLink

Abstract

From some year ago, the drone's sector is having a strong evolution and the number of companies that manufacture these devices are increasing constantly. Consequently the price of them are becoming cheaper every day, because of this price reduction the companies and public en general can allow themselves to buy more than one drone.

In these circumstances and due to the majority of companies design their own system to communicate with its drones, this creating the difficulty of having to learn to use each of them independently.

In order to solve this problem in this project has been developed an application which use the MAVLink protocol to control different models of drones with a common interface for all of them.

This functionality by itself is of great utility, but also with the objective of achieving an application with a major commercial potentiality, in this project it has been included the development of two additional functionalities: the possibility of making simultaneous flights with more than one dron at the same time, and also realize the communication with drones through an intermediate proxy to obtained major versatility.

Keywords: Dron, MAVLink

Índice general

Índice de figuras

Índice de tablas

Glosario

SMS	*Short Message Service*
RPV	*Remotely Piloted Vehicle*
UAV	*Unmanned Aerial Vehicle*
RPA	*Remotely-Piloted Aircraft*
RPAS	*Remotely-Piloted Aircraft System*
OACI	*Organización de Aviación civil Internacional*
GPS	*Application Service Provider*
GCS	*Ground Control Stations*
XML	*eXtensible Markup Language*
LGPL	*GNU Lesser General Public Licencense*
CRC	*Cyclic Redundancy Check*
CAN	*Controler Area Network*
SAE	*Society of Automotive Engineers*
ITU	*International Telecommunication Union*
IEEE	*Institute of Electrical and Electronics Engineers*
JVM	*Java Virtual Machine*
URL	*Uniform Resource Locator*
RAM	*Random Access Memory*

Capítulo 1

Introducción y objetivos

1.1 Introducción

En estos días de todos es conocida la velocidad con la que avanza la tecnología. Pero si nos fijamos en los últimos años, se está avanzando tan rápido, que es muy difícil estar al día de las nuevas tecnologías o los avances en las ya existentes.

Un buen ejemplo para observar este comportamiento son los teléfonos móviles. Desde la llegada de estos dispositivos al usuario común, se fueron realizando avances de manera progresiva, sobre todo centrándose en la reducción de tamaño de los dispositivos, mejorar la cobertura y agregando funcionalidades enfocadas a la mejora de la comunicación como los SMS. Pero en los últimos años, desde la llegada de los llamados móviles inteligentes, la velocidad con que se están produciendo cambios es impresionante. De esta manera hoy en día, podemos disponer de terminales que hacen fotografías como una cámara compacta, reconocen nuestras huellas dactilares o se pueden sumergir en el agua, pero sobre todo, son incluso más potentes que ordenadores de hace algunos años.

Otro dispositivo de alto carácter tecnológico que está teniendo una gran evolución son los denominados drones. En los últimos años la comercialización y popularidad de estos aparatos se ha incrementado de una manera impresionante y, en consecuencia, se

están reduciendo sus costes, creando muchas posibilidades de las que antes no se disponía. Pero la evolución de este sector no tiene los mismos objetivos que el de los teléfonos móviles, donde el desarrollo de la tecnología se ha centrado en satisfacer las necesidades del público general. Los drones, por otra parte, además del uso recreativo que le puede dar una persona común, tiene muchas más posibilidades, tanto en el ámbito comercial como en la seguridad civil.

En el ámbito comercial los drones pueden ser usados, por ejemplo, para controlar mejor la seguridad de un recinto, entregar paquetes de una manera más rápida, monitorear grandes extensiones de terreno cultivado o fumigar dichos terrenos. Pero también puede ser de gran ayuda para la seguridad civil, en la prevención y control de incendios, control de fronteras o en el salvamento de personas.

Como se puede ver, el uso de los drones tiene muchas posibilidades y el mercado se ha dado cuenta de ello, por lo cada vez hay más empresas que se dedican a fabricarlos o a crear productos, software o sistemas para ellos. Por no hablar de otros centros de investigación, como por ejemplo las universidades, que también están realizando muchos proyectos con ellos en los últimos años.

1.2 Motivación

Como se ha comentado anteriormente, la rápida evolución del sector está generando que cada vez más empresas diferentes se dediquen a fabricar drones. Esto tiene un lado positivo, los aparatos están evolucionando muy rápido y se están encontrando muchas nuevas funcionalidades y usos para ellos. Pero también tiene un lado negativo, como pasa con las tecnologías que tienen un rápido crecimiento, esta velocidad va en contra de la homogeneidad de los productos obtenidos.

Si mañana decidimos comprarnos un dron en el mercado vamos a encontrar muchos dispositivos, tanto si lo queremos utilizar para uso recreativo como para uso profesional, con un gran abanico de precios que dependen de las funcionalidades, sensores o de la estabilidad y fiabilidad de vuelo que tengan dichos aparatos. Pero aquí surge un problema, normalmente cada empresa crea sus propios sistemas y protocolos para realizar el control remoto y hasta puede existir el caso que una empresa tenga más de uno para sus diferentes modelos. Si una persona, por necesidad o gusto, dispone de más de un dron, tendrá que aprender a manejar cada uno de manera independiente.

Debido a que probablemente el mercado va seguir evolucionando como hasta ahora, por lo menos durante unos años más, se ve la necesidad de buscar una solución para que el usuario final no tenga que tener que aprender a manejar cada dron nuevo que se compre.

1.3 Objetivos

Con el desarrollo de este proyecto queremos crear una aplicación para ordenador desde la cual se puedan manejar diferentes modelos de drones. Además, ya que vamos a crear una aplicación donde se pueda configurar más de un dron, se nos presenta la posibilidad de crear vuelos simultáneos con ellos, una funcionalidad que puede resultar muy útil.

Se fijan a continuación los requisitos que debe que cumplir la aplicación para que pueda considerarse una solución final del proyecto aceptable.

Requisitos principales:

- Control de drones: desde la aplicación se debe poder controlar diferentes modelos de drones o que sean de diferente compañía.

Requisitos secundarios:

1. Vuelo simultáneo: si la aplicación tiene configurado más de un dron se debe poder realizar vuelos simultáneos donde cada uno de ellos se maneje de manera independiente.
2. Proxy: añadir un proxy intermedio en la comunicación con los drones que podamos colocar en otro dispositivo. Con esto podemos conseguir, por ejemplo, ampliar la cobertura de control de los drones.

1.4 Estructura de la memoria

Este documento está formado por 9 capítulos que vamos a describir brevemente para facilitar la lectura del mismo.

- **Introducción y objetivos:** se realiza una breve introducción al problema existente para controlar varios drones y se fijan unos objetivos para poder solucionarlo.
- **Definición e historia:** en este capítulo se explican los diferentes nombres dados a los drones y se presenta un corto resumen de la evolución de los drones a lo largo historia.
- **Estado de arte:** se analiza la evolución actual del sector de los drones. En este capítulo podemos diferenciar tres partes: la primera que describimos brevemente alguno de los usos que se le pueden dar a los drones y los tipos de drones existentes; la segunda donde se analizan los sistemas utilizados para controlar drones; y la última donde estudia la normativa que regula el uso de estos dispositivos en España.
- **MAVLink:** se estudia el protocolo MAVLink y se describen las parten de las que está compuesto.

- **Diseño:** en este capítulo se presentan las decisiones que se han tenido que realizar a la hora de diseñar la aplicación.
- **Implementación:** durante este capítulo se describe como se ha realizado la programación de la aplicación para cumplir los objetivos y seguir el diseño marcado en el capítulo anterior.
- **Validación:** se presentan las pruebas realizadas para comprobar el perfecto funcionamiento de la aplicación y que se cumplan los objetivos que nos hemos propuesto. Además se describen los drones y otros dispositivos que hemos elegido para realizarlas.
- **Planificación y entorno socio-económico:** en este capítulo realiza una descripción de la metodología usada para realizar el proyecto y se describen las fases que se han seguido durante su desarrollo. Además se presenta un presupuesto de los costes del proyecto y el posible impacto socio-económico que se obtiene con su finalización.
- **Conclusiones y trabajos futuros:** en el capítulo final del proyecto se presentan las conclusiones obtenidas durante su desarrollo y se describen algunos futuros trabajos que se pueden realizar para ampliar la aplicación creada como resultado del mismo.

Capítulo 2

Definición e historia

2.1 Introducción

En este capítulo se va a presentar el origen del término dron y los diferentes nombres que las aeronaves no tripuladas han tenido a lo largo de la historia. Además se resumirá brevemente la evolución de los drones en la historia y cuáles han sido sus usos durante esta.

2.2 Definición

La palabra dron, procede del término inglés *drone*, que significa zángano o zumbido. Hay varias teorías sobre la razón por la que se les dio ese nombre a los aviones no tripulados, algunos piensan que se debe al sonido que emiten los motores, asemejándose a un zumbido; otros, como el historiador Steven Zaloga [TFD], comentan que el origen se produce en 1935 cuando un almirante de los Estados Unidos vio una demostración británica de su nuevo avión no tripulado, llamado *Queen bee*, y cuando volvió a su país pidió que se crearan dispositivos del mismo tipo, denominándolo *drone* en homenaje al

original. De todas formas, sea cual sea el origen real, todas las teorías coinciden que es de origen militar.

El nombre que se le ha dado a los aviones no tripulados ha ido evolucionando con el tiempo. Durante la guerra de Vientan se les llamaba RPV, unas siglas cuyo significado original es Remotely Piloted Vehicle. En la década de los 90 se popularizo el término UAV (Unmanned Aerial Vehicle), siendo el Ministerio de Defensa de los Estados Unidos, mediante la publicación de un diccionario [DDD], quien delimito los dispositivos que entraban en esta categoría en la siguiente definición:

Un vehículo aéreo motorizado que no lleva a bordo a un operador humano, utiliza las fuerzas aerodinámicas para generar la sustentación, puede volar autónomamente o ser tripulado de forma remota, que puede ser fungible o recuperable, y que puede transportar una carga de pago letal o no. No se consideran UAV a los misiles balísticos o semibalísticos, misiles crucero y proyectiles de artillería.

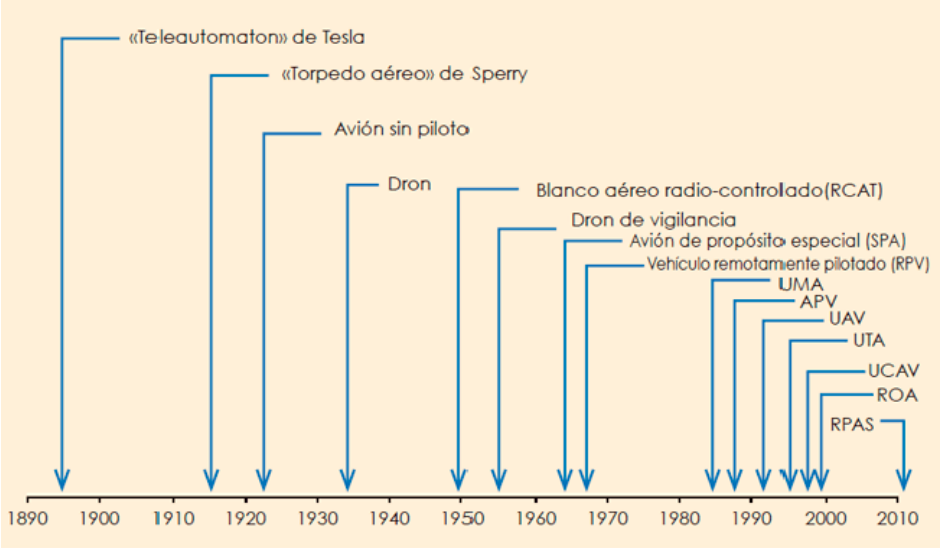

Figura 1. Términos que han designado a los drones a largo de la historia
(Fuente: drones.uv.es)

Como se ve en la figura 1, hay más nombres que podríamos destacar, pero el único término que se considera correcto actualmente es el último RPAS. Este término fue creado en el año 2011, cuando la organización de Aviación civil Internacional (OACI) reconoció por primera vez a los vehículos aéreos no tripulados como aeronaves y acuño los siguientes términos:

- RPA, Remotely-Piloted Aircraft: una aeronave en la que el piloto al mando no está a bordo.
- RPAS, Remotely-Piloted Aircraft System: un conjunto de elementos configurables formado por un RPA, su estación de pilotaje remoto asociada (RPS – Remote Pilot Station), el sistema requerido de enlace de mando y control y cualquier otro elemento requerido en cualquier punto durante la operación del vuelo.

Aunque el término dron ya no sea el adecuado para referiste a estos dispositivos, se seguirá usando en el resto del texto debido a que es por el que más comúnmente se los conoce.

2.3 Historia

Los vuelos no tripulados se llevan usando mucho tiempo en las guerras. Mucho antes de que se empezara a pensar en algo remotamente parecido a lo existente actualmente, en julio de 1849 está registrado el primer ataque aéreo con un dispositivo de este tipo. Se realizó mediante globos aerostáticos controlados por cuerdas y cargados de explosivos. Claro que no podemos denominar a estos globos como drones, pero con esto queda claro lo interesados que siempre han estado los militares en estos dispositivos.

Pasaron unos años hasta que se realizase una nueva innovación que nos acercase a algo parecido a los aparatos actuales. En 1898 se empezó a practicar la primera vigilancia aérea, durante la Guerra Hispano-Americana, cuando los militares de Estados Unidos instalaron una cámara en una cometa. Estas cometas se siguieron utilizando durante la Primera Guerra Mundial

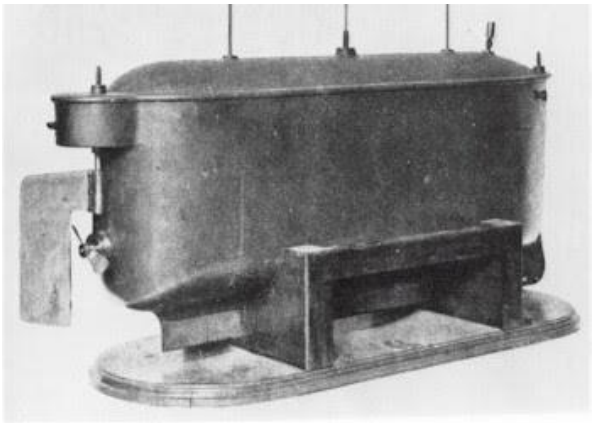

Figura 2. Teleautomaton (Fuente: Wikipedia)

Durante esa misma época Nikola Tesla creo el *Teleautomaton*, un artefacto naval capaz de avanzar, pararse, ir a derecha o izquierda y hacer destellar sus luces mientras enviaba diferentes frecuencias de radio. Este aparato se puede considerar el uno de los precursores reales de los misiles cruceros y del concepto actual de la aviación no tripulada.

Ya durante la Primera Guerra Mundial, Elmer Ambrose Sperry ayudo al avance del control remoto y la navegación automática, gracias al desarrollo de un giroestabilizador que se había inventado en 1909. Durante el resto de la guerra se siguieron desarrollando torpedos aéreos no tripulados.

En la Segunda Guerra Mundial el gobierno de Gran Bretaña creó el blanco aéreo con control completo por radio llamado *Queen bee*. Mientras tanto en Estados Unidos se desarrolló el *RP4*, que se produjo como sistema de entrenamiento. A través de estos aviones, se fue desarrollado de manera temprana la tecnología y el uso de control remoto por radio.

Después de la guerra, el desarrollo de los drones se ralentizo debido a la poca necesidad de ellos y del éxito que se estaba obteniendo en el campo de los misiles. De todas formas cabe destacar los sistemas *Crossbows*, cuya funcionalidad era lanzarlos desde los bombarderos y servir como señuelo de radar.

Con la guerra de Vietnam se volvió a impulsar el desarrollo de los aviones no tripulados. Pero esta vez el objetivo no fue crear bombas, si no que se intentaba crear un programa de vigilancia. Durante esta época, los drones ya eran más perecidos a los que conocemos hoy en día, el sistema de radiocontrol estaba mucho más avanzado, contaban con GPS y muchos sensores, además también se empezó a añadir el sistema de propulsión a reacción.

En la década de los 70 se afianzo el uso de drones para la vigilancia aérea, creando aparatos diseñados para misiones de reconocimiento y vigilancia tanto de corto alcance, como de largo alcance y elevada altitud.

Sistemas no tripulados más relevantes en la década de los 80.
De izquierda a derecha y de arriba abajo, el *Lockheed Aquila*, el MBLE *Epervier*, el *Westland Wisp* y el *Compass Cope*.

Figura 3. Drones de la década de los 80 (Fuente: drones.uv.es)

Durante los siguientes años se siguió mejorando los aviones no tripulados pero no fue hasta la década de los 90, con la mayor disponibilidad del GPS a nivel mundial y de las comunicaciones por satélite, cuando se pudo empezar a volar fuera del limitado alcance que ofrecía la comunicación por radio. Fue al final de esta década cuando Estados Unidos empezó a dotar de misiles a algunos de sus aparatos.

No es hasta este siglo cuando se empiezan usar drones de manera civil. Pero seguían siendo sistemas muy caros y solo se usaban en grandes proyectos, por ejemplo, la NASA los empezó a usar para analizar las capas altas de la atmosfera. No obstante, el mercado

siguió avanzado, dando un gran salto en los últimos años, cuando se han abaratado los costes y posibilidad de acceso al público común.

Capítulo 3

Estado de arte

3.1 Introducción

En este capítulo se realiza un análisis de la evolución actual del sector de los drones. Este análisis consta de tres partes: la primera donde se estudia el uso de los dispositivos en el mercado y se realiza una explicación de los tipos de drones existentes; la segunda donde nos centramos en las aplicaciones existentes para el control de drones; y una tercera donde se describe la normativa aplicable a estos aparatos.

3.2 Usos de los drones

Como ya se ha explicado el uso y evolución de los drones a lo largo de la historia ha sido en su mayoría con objetivos militares, pero en los últimos años esto ha cambiado. Actualmente, con la reducción del precio de los aparatos, se está creando un mercado que puede fabricar dispositivos y funcionalidades que solucione o facilite problemas comerciales o de seguridad civil. Podemos citar algunos ejemplos que ratifican el por qué muchas empresas e instituciones tienen tanto interés y realizan investigaciones sobre los drones.

- Mensajería: con los drones se puede reducir el tiempo que se tarda en entregar un paquete, pero también es útil para llegar de manera más sencilla y barata a zonas menos urbanas donde en general no hay rutas comerciales.
- Grabación de eventos: antes solo se podía grabar grandes eventos donde ya se dispusiese de cámaras fijas. Con los drones es más fácil y barato, no se necesitan tantas "cámaras" y se puede cambiar la localización de las mismas dependiendo de la situación.
- Agricultura: permite a los agricultores monitorear grandes extensiones de terrenos, reduciendo horas de trabajo y personal. Además pueden ayudar a fumigar dichos terrenos.
- Investigaciones biológicas: se puede localizar y seguir a especies animales en su ambiente natural de una manera mucho más eficiente que por otros medios.
- Inspección de estructuras: cualquier tipo de estructura requiere un mantenimiento a lo largo de los años. Esto es especialmente prioritario en estructuras peligrosas como las centrales nucleares o los puentes. Con los drones se facilita mucho la comprobación de fallas y elimina riesgos para las personas.
- Control de incendios: estos aparatos son muy útiles para este uso, ya que pueden ayudar en la prevención, vigilando los estados de los bosques, como durante los propios incendios, ayudando a su eliminación echando agua o productos químicos desde el aire, observando la extensión del incendio, hacia donde se dirige o como estación meteorológica.
- Salvamento de personas: los drones pueden recorrer grandes extensiones de terreno de una manera muy rápida, también pueden llegar a sitios que por otros medios no se podría. Además que pueden localizar a la gente de otras maneras, como pasa con el proyecto LifeSeeker [Lif], donde los móviles de las personas a salvar emiten una señal que puede ser localizada por el dron.
- Llegar a zonas peligrosas: estos dispositivos pueden entrar en volcanes y recoger muestras o, como pasó en Fukushima, donde se utilizaron drones para entrar en el rector nuclear.

3.3 Tipos de dron

Como se puede ver, los drones se pueden utilizar para una gran diversidad de funciones y es muy difícil que un solo tipo se ajuste a todas ellas. Y, aunque el mercado se adapta, innova y crea aparatos diferentes según sea el uso que se le pueda dar, la mayoría de los drones tienen una morfología común que se puede clasificar de la siguiente manera:

- Ala fija: este dron es el que mejor autonomía tiene debido a que es el más eficiente aerodinámicamente hablando y puede estar más tiempo planeando

que volando. Es el más idóneo para vigilancias prolongadas y de gran extensión. Pero esta morfología tiene desventajas importantes como la necesidad de disponer de una extensión dedicada como pista de aterrizaje o, al disponer de solo motores traseros, no puede mantenerse parado en el aire y realizar cambios muy bruscos en la dirección del vuelo.

Figura 4. Dron de ala fija (Fuente: dronespain.pro)

- Multirrotor: son los más populares hoy en día, tienen gran versatilidad y eficacia. Son bastante estables debido a que todos sus motores se encuentran a la misma distancia del centro de gravedad del dispositivo. Pueden tener diferente número motores, cuanto más tengan más estable va ser. El punto negativo de este tipo de dron es su baja autonomía debido al gran gasto de energía que requiere hacer funcionar un mayor número de motores.

Figura 5. Dron tipo multirrotor (Fuente: dronespain.pro)

- Helicóptero: no hace falta describir la forma de este dron. Son los más polivalentes, estando en un punto medio entre los multirrotores y los drones de ala fija.

Figura 6. Dron tipo helicóptero (Fuente: dronespain.pro)

3.4 Control de drones

Uno de los problemas del crecimiento tan rápido de un mercado donde existen tantas aplicaciones y tipos de drones diferentes, es que no se consigue productos finales estandarizados y, normalmente, cada compañía crea sus propios protocolos y sistemas de control remoto.

Debido a esto, programadores entusiastas de estos dispositivos, empezaron hace algunos años a crear aplicaciones dedicadas al control genérico de diferentes tipos de dron. Y han continuado trabajando en ello, dejando atrás la simple funcionalidad del manejo manual y llegando a la tendencia actual, el piloto automático o *autopilot* como se denomina en inglés.

Estos sistemas de pilotaje están constituidos por tres partes diferenciadas:

1. Estación de control o Ground Control Stations (GCS): es el centro de control desde el cual podemos comunicarnos y controlar los drones.
2. Piloto automático: es el hardware y código que controla que el vuelo se mantenga en los parámetros de la ruta establecida por la estación base.
3. Protocolos: son protocolos para la creación de mensajes que sirven de comunicación entre las estaciones de control y los drones donde está el firmware de piloto automático.

3.4.1 Estación de control

Se denomina estación de control a todo el conjunto de instalaciones y software desde el cual se puede controlar a un dron. Estas instalaciones tienen que disponer de todo el hardware necesario para poder comunicarnos con los drones en todo momento y poder recibir la telemetría constantemente, considerando básicos: un ordenador; una interfaz hombre-máquina; un receptor y emisor de radio o equivalente. Además debe disponer de algún software que permita la comunicación y el control del dron en todo momento.

En el caso de los sistemas de piloto automático, además del sistema de control, el software tiene que ser capaz de crear una misión en la que se describirá la ruta que debe seguir el dron en un mapa.

Actualmente hay muchas aplicaciones de software libre que tienen este propósito, algunas son específicas para un sistema de piloto automático concreto, pero la mayoría son comunes para la mayoría de los sistemas. De entre todas podríamos destacar las siguientes:

- Mission Planner: es un ejemplo de estación de control diseñado específicamente para un solo piloto automático, ArduPilot. La aplicación se creó en 2010, es decir, que lleva casi desde el origen de la revolución de los drones en el ámbito civil, y ha ido evolucionando desde entonces, por lo que es una de las más estables y con funcionalidades del mercado. El único problema es que solo está programado para sistemas operativos Windows y para Mac OS X utilizando Mono.
- APM Planner 2.0: es la solución adoptada por el grupo de desarrolladores de ArduPilot para los usuarios de Linux o Mac OS; también se puede instalar en ordenadores con sistema operativo Windows, pero no es el objetivo principal de la aplicación y al no ser la estación de control principal del sistema no dispone de tantas funcionalidades como Mission Planner.
- QGroundControl: esta aplicación la ha creado el grupo de desarrollo de MAVLink, por lo que es una aplicación que se puede usar en todos los sistemas que entiendan este tipo de mensajes. Tiene una interfaz bastante sencilla y clara, pero con pocas funciones complejas. Es una de las aplicaciones que está programada para más sistemas operativos, funcionando en Windows, Mac OS, Linux, Android e iOS.

Figura 7. Aplicación QGroundControl (Fuente: qgroundcontrol.com)

- MavProxy: al igual que en el caso anterior esta creada por el grupo de desarrollo en MAVLink, pero en este caso el objetivo es crear la estación de control más ligera y portable posible. Está escrita en el lenguaje de programación Python, por lo que puede funcionar en cualquier sistema operativo que lo entienda. Para reducir lo máximo posible el tamaño, la aplicación no dispone de interfaz gráfica, dependiendo totalmente de una terminal de líneas de comando. La aplicación se puede ampliar mediante módulos.

3.4.2 Piloto automático

Es la parte encargada de controlar el vuelo del dron para que siga los parámetros establecidos en la misión creada por la estación de control, es decir, tiene que ser capaz, por ejemplo, de cambiar de dirección al llegar a un determinado punto, ascender o descender para mantener siempre la misma altura respecto al suelo o corregir el vuelo si por el viento o por otro factor se ha desviado de la ruta original. Para poder realizar estos cambios en el vuelo se necesitan dos cosas: un software que controle el estado de la misión y mande los comandos necesarios para que se siga esta, y un hardware donde se instale el software anterior y que se conecte con los componentes y sensores del dron.

De esta parte del sistema hay muchas menos opciones de software libre que en el caso anterior, pudiendo destacar claramente dos:

- ArduPilot: se le puede considerar el piloto automático más utilizado del mercado. Esto se debe principalmente, a que fue de los primeros en desarrollarse y ha continuado evolucionando desde entonces, creando una comunidad entorno a él cada vez más grande. Actualmente es uno de los software más estables y con mayor número de funcionalidades, permitiendo no solo controlar drones aéreos, sino que también permite controlar coches, barcos o submarinos teleredigidos.

- PX4: es otro piloto automático bastante usado por los usuarios, pero no dispone de tantas funcionalidades como ArduPilot y solo puede controlar drones aéreos.

Como se ha comentado, además de un software para controlar la misión y se necesita un hardware donde instalarlo. En el mercado hay muchas placas integradas con este fin, pero las más utilizadas por los usuarios son las que desarrollan los propios grupos que crean el software de piloto automático que vayan a utilizar, dando como resultado la preferencia por las placas APM (ArduPilotMega) y PIXHAWK, desarrollados por los grupos de ArduPilot y PX4. Ambas placas integradas son bastante económicas, pero PIXHAWK, tiene la ventaja de que soporta tanto el software de ArduPilot como el de PX4.

3.4.3 Protocolos

Para poder cargar la misión en el dron desde la estación base y además poder recibir la telemetría o mandar comandos de urgencia que no estén en la misión original, hace falta un protocolo de comunicación fiable y rápido entre ambos puntos.

Existen muy pocos protocolos de comunicaciones para drones, siendo MAVLink el utilizado en la mayoría de los casos. Pero esto no quiere decir que no existan otros, las empresas privadas de drones comerciales normalmente desarrollan los suyos propios, y también existen otros software de uso libre como por ejemplo el protocolo UAVCAN desarrollado por el MIT.

3.4.3.1 MAVLink

MAVLink es un protocolo diseñado específicamente para realizar la comunicación entre las estaciones de control y los drones y, como su propio nombre indica (Micro Air Vehicle Communication Protocol), está concebido por sus desarrolladores para poder intercambiar información solo con vehículos aéreos.

Cuando se creó este protocolo en 2009, se buscaba una forma de comunicación sencilla, ligera y fiable, con el objetivo de que pudiese ser integrado fácilmente en cualquier tipo de dron, teniendo en cuenta que estos pueden tener una capacidad hardware limitada. Otro de los objetivos que se buscaba era enviar el mínimo contenido posible por radio, reduciendo así el ancho de banda usado en la comunicación. El resultado fue una librería muy ligera que contiene toda la información sobre los mensajes utilizados en la comunicación con micro vehículos aéreos.

MAVLink es utilizado actualmente por muchos tipos de drones del mercado, esto posiblemente se deba a que, desde su origen, se planteó crearlo de una manera que no fuera solamente ligera, sino que también se pudiera implementar muy fácilmente en diferentes sistemas. Los mensajes que se utilizan para la comunicación están definidos en XML (eXtensible Markup Language), por lo que es muy fácil manejarlos con diferentes lenguajes de programación. Además de ello MAVLink cuenta con una licencia LGPL (GNU Lesser General Public Licencense), que permite ser usado tanto para crear proyectos open-source (libres) como close-source (privados), lo que ha permitido que pueda ser usado por empresas con poco presupuesto que no pueden invertir en crear su propio protocolo.

Pero MAVLink también tiene factores negativos, siendo el principal la seguridad. Para hacerlo lo más ligero posible, el protocolo solo cuenta con una simple comprobación de errores del tipo CRC y no implementa ningún sistema más complejo para evitar la manipulación externa, siendo muy sencillo interferir entre la comunicación del dron y la estación de control.

3.5 Marco regulador

En este apartado se va a describir la legislación existente sobre el uso de drones en el territorio español. Para hacerlo de manera auto-contenido, se ha preferido realizar un resumen en vez de hacer una referencia externa de toda ella.

3.5.1 Introducción

Debido al gran y rápido crecimiento del uso de drones, el gobierno de España se vio en la necesidad de redactar una normativa temporal que regulará su uso en espacio aéreo de su territorio. Esta regulación que se recogió en el artículo 50 del Real Decreto-ley 8/2014, del 4 de Julio de 2014, para luego convertirse en ley el 17 de octubre de 2014, Ley 18/2014.

En la propia ley se indica que este marco regulatorio es temporal y actualmente hay un borrador de un Real Decreto en el que se desarrolla la regulación completa y sustituirá al marco actual cuando se apruebe.

Además de esta regulación específica, también tienen que cumplir la regulación general de navegación aérea, recogida en la Ley 48/1960 y en el Real Decreto 552/2014.

3.5.2 Drones

Los drones deberán estar inscritos en el Registro de matrícula de aeronaves y tener un certificado de aeronavegabilidad emitido por la Agencia Estatal de Seguridad Aérea, exceptuando las que en el momento del despegue tengan una masa igual o inferior a 25 kilogramos.

De todas formas, todas drones tienen que llevar fijada a su estructura una placa de identificación en la que deberá constar de forma legible a simple vista e indeleble la siguiente información:

- La identificación de la aeronave, mediante una designación específica y, en su caso, número de serie.
- El nombre de la empresa operadora de la aeronave.
- Los datos necesarios para ponerse en contacto con la operadora de ser necesario.

3.5.3 Condiciones de vuelo

Para poder realizar vuelos con drones será necesario que haya condiciones meteorológicas visuales y de día.

Según las características de la aeronave podrá operar de la siguiente manera:

- Las aeronaves que pesen menos de 2 kilogramos en el momento del despegue deberán operar fuera de aglomeraciones de edificios, pueblos, lugares habitados o de reunión de personas al aire libre, en espacio aéreo no controlado, más allá del alcance visual del piloto, siempre que esté al alcance de la emisión de radio y vuele a una altura máxima de 400 pies (120 metros). También tendrá que llevar el equipamiento necesario para poder ser localizado en cualquier momento y poder enviar un NOTAM [Not].
- Las aeronaves que pesen menos de 25 kilogramos en el momento del despegue deberán operar fuera de aglomeraciones de edificios, pueblos, lugares habitados o de reunión de personas al aire libre, en espacio aéreo no controlado y dentro del alcance visual del piloto, a una distancia horizontal no mayor de 500 metros y a una altura máxima de 400 pies (120 metros).
- Las aeronaves que pesen en el momento del despegue entre 25 y 150 kilogramos y aquellas cuya masa máxima supere los 150 kilogramos destinadas a la realización de actividades de lucha contra incendios o búsqueda y salvamento, solo podrán operar según las condiciones establecidas en su certificado de aeronavegabilidad.

3.5.4 Actividades

La legislación actual y más concretamente la Ley 18/2014, regula las aeronaves civiles para su uso comercial o profesional, por lo que quedan excluidos de este marco regulatorio los vuelos con fines recreativos o deportivos.

Las tipos de vuelo que se contemplan en esta ley son los siguientes:

- Vuelos de prueba de producción y de mantenimiento, realizados por fabricantes u organizaciones dedicadas al mantenimiento.
- Vuelos de demostración no abiertos al público, dirigidos a grupos cerrados de asistentes a un determinado evento o de clientes potenciales de un fabricante u operador.
- Vuelos para programas de investigación, nacionales o europeos, en los que se trate de demostrar la viabilidad de realizar determinada actividad con drones.
- Vuelos de desarrollo en los que se trate de poner a punto las técnicas y procedimientos para realizar una determinada actividad con drones previos a la puesta en producción de esa actividad.
- Vuelos de I+D realizados por fabricantes para el desarrollo de nuevos productos.

- Vuelos aéreos, fitosanitarios y otros que supongan esparcir sustancias en el suelo o la atmósfera, incluyendo actividades de lanzamiento de productos para extinción de incendios.
- Vuelos de observación y vigilancia aérea incluyendo filmación y actividades de vigilancia de incendios forestales.
- Vuelos para realizar publicidad aérea, emisiones de radio y TV.
- Operaciones de emergencia, búsqueda y salvamento.

3.5.5 Requisitos de vuelo

Para poder realizar cualquiera de las actividades antes mencionadas es necesario que se cumplan los siguientes requisitos:

- El operador debe disponer de la documentación relativa a la aeronave, incluyendo la definición de su configuración, características y prestaciones.
- Un manual de operaciones donde el operador establezca los procedimientos de la operación.
- Un estudio aeronáutico de seguridad de la operación.
- Haber realizado los vuelos de prueba que resulten necesarios para demostrar que la operación se puede realizar con seguridad
- La aeronave tiene que tener un plan de mantenimiento que se ajuste a las recomendaciones del fabricante.
- La operación se tiene que realizar por pilotos certificados.
- Se debe disponer de una póliza de seguro u otra garantía financiera que cubra la responsabilidad civil frente a daño a terceros.
- Se deben adoptar las medidas adecuadas para proteger la aeronave de actos de interferencia ilícita durante las operaciones.
- Tienen que existir las medidas adicionales necesarias para garantizar la seguridad de la operación y para la protección de personas y bienes subyacentes.

Además de cumplir con todos estos requisitos es necesario informar y contar con la aprobación de la Agencia Estatal de Seguridad Aérea antes de poder realizar cualquier vuelo.

3.5.6 Pilotos

Los pilotos deben acreditar los siguientes requisitos para poder realizar operaciones autorizadas con drones:

- Se debe disponer de uno de los términos siguientes:
  - Ser titulares de cualquier licencia de piloto, incluyendo las licencias de ultraligero, emitida conforme a la normativa vigente, o haberlo sido en los últimos cinco años.
  - Demostrar de manera fehaciente que se tiene de los conocimientos teóricos necesarios para obtener cualquier licencia de piloto, incluyendo las licencias de ultraligero.
  - Para drones que pesen menos de 25 kg que vuelen dentro del alcance visual del piloto, un certificado básico de pilotaje de aeronaves emitido por una organización de formación aprobada.
  - Para drones que pesen menos de 25 kg que vuelen fuera del alcance visual del piloto, un certificado avanzado de pilotaje de aeronaves emitido por una organización de formación aprobada.
- Tener los 18 años cumplidos.
- Se deberá disponer de uno de los siguientes certificados médicos:
  - Para drones que pesen menos de 25 kg, se debe disponer de certificados médicos para la licencia de piloto de aeronave ligera (LAPL).
  - Para drones que pesen más de 25 kg, los pilotos deberán ser titulares de un certificado médico de clase 2.
- Se debe disponer de un documento que acredite que se disponen de los conocimientos adecuados de la aeronave y sus sistemas, así como su pilotaje, emitido por el operador.

3.5.7 Reglamento para vuelos recreativos

Como se ha dicho anteriormente la Ley 18/2014 no regula los vuelos deportivos o recreativos de aeronaves pilotadas por control remoto. Pero esto no significa que cuando se realicen estas actividades no haya ninguna norma que seguir.

La Agencia Estatal de Seguridad Aérea informa que en estos casos se tiene que seguir la regulación que tenga cada Comunidad Autónoma o Municipio, aunque siempre hay que respetar la legislación aeronáutica general.

De todas formas la Agencia Estatal de Seguridad Aérea realiza las siguientes recomendaciones para este tipo de vuelo:

- No volar sobre centros urbanos o sobre personas.
- No volar cerca de aeropuertos.
- No volar de noche.
- Siempre tener a la vista el dron y a una altura máxima de 120 metros.

También avisa de que el uso indebido del dron que cause algún daño o perjuicio a terceros puede acarrear multas de hasta 225.000 euros.

3.5.8 Nuevo marco regulador

Como he indicado al principio de este capítulo, esta ley es provisional hasta que se apruebe un marco regulador más completo. Actualmente hay un borrador de un Real Decreto del 27 de octubre de 2017, en el que realiza una regulación más detalla de la normativa descrita en la Ley 18/2014.

De este borrador podemos destacar los siguientes puntos:

- El piloto debe poder intervenir en el control del dron en cualquier momento del vuelo.
- El sistema de control del dron tiene que garantizar la ejecución de la funciones con continuidad y fiabilidad, siempre respetando la normativa vigente en relación con el uso del espacio radioeléctrico.
- Se amplían las condiciones de vuelo permitiendo los siguientes vuelos:
    - Para aeronaves que no dispongan de certificado de aeronavegabilidad se podrán hacer vuelos con las mismas características que antes, pero ahora se ha aumentado la distancia de vuelo mediante la introducción de observadores. Estos observadores tendrán que cumplir los mismos requisitos que el piloto y estar en contacto permanente por radio con él. Durante el vuelo el dron podrá estar a una distancia máxima de 500 metros del piloto o de los observadores.
    - Para el vuelo en el que no hace falta estar en contacto visual con el dron se agrega, además de los drones de 2 kilogramos, a aquellos que cuenten con sistemas, aprobados por la Agencia Estatal de Seguridad

Aérea, que permitan al piloto detectar y evitar a otros usuarios del espacio aéreo.

  - Se permite el vuelo en zonas pobladas a drones que no exceda los 10 kg al despegue, siempre que sean zonas acotadas, en las que la autoridad competente haya limitado el paso a personas y vehículos o, en todo caso, se mantenga una distancia mínima de 150 metros con respecto a edificios y de 50 metros respecto a personas.

- Se prohíbe el pilotaje de drones desde vehículos en movimiento, a menos que se cuente con un plan operacional de vuelo que garantice el constante manejo de los mismos.
- Se prohíbe el control de más de un dron al mismo tiempo.

Capítulo 4

MAVLink

4.1 Introducción

En este capítulo se va a tratar de analizar el protocolo MAVLink para poder comprender su funcionamiento e integrarlo en la solución final para comunicarse con los drones.

MAVLink es un protocolo cuya finalidad es la realización de una comunicación con drones donde se use el menor ancho de banda posible. El protocolo se basa en una librería muy ligera de mensajes definidos en XML, que es fácilmente integrable en la mayoría de los drones debido al poco espacio que ocupa. Además de ello los creadores del protocolo han optimizado una implementación en lenguaje de programación C/C++, para sistemas con pocos recursos. Debido a todos estos factores hay mucha gente que se ha interesado en este protocolo y lo ha implementado en sus drones, siendo usado actualmente por un gran número de dispositivos en el mercado actual.

4.2 Paquete

La comunicación en el protocolo MAVLink se basa en mensajes que se encapsulan en una secuencia codificada de bytes para dar forma a un paquete. Dicho paquete tiene una estructura compuesta por una cabecera, un payload (contenido útil) y un checksum (suma de verificación), que está inspirada en los protocolos CAN bus (Controller Area Network) y SAE AS-4.

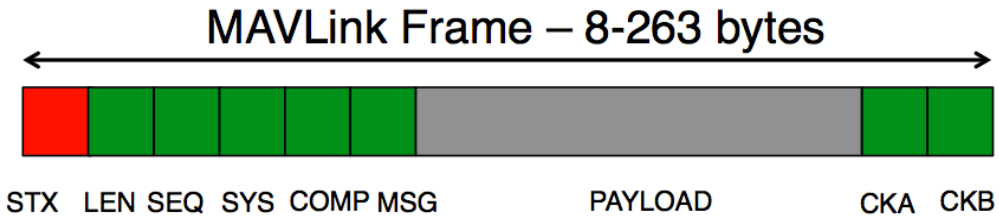

Figura 8. Paquete MAVLink

En la figura superior se puede observar más detalladamente la estructura del paquete, en el que cada recuadro, menos en el payload, representa un byte con la siguiente información:

- Cabecera: Está compuesta por los 6 bytes que preceden al payload
  - STX: este byte indica el inicio del paquete MAVLink. Este byte siempre tiene el mismo valor y no hay que modificarlo.
  - LEN: en este byte se especifica el tamaño que tiene la carga útil del mensaje, es decir, el payload. Puede tomar un valor entre 0 y 255.
  - SEQ: MAVLink permite numerar el paquete antes de enviarlo. Esto puede servir, por ejemplo, para reordenar un mensaje de un tamaño tan grande que hace falta enviarlo en más de un paquete. Puede tomar un valor entre 0 y 255.
  - SYS: la información que contiene este byte indica el identificador del sistema emisor. Gracias a este identificador podemos diferenciar diferentes dispositivos en una misma red. Puede tomar un valor entre 1 y 255, siendo este último comúnmente asignado a la estación de control.
  - COMP: un dron puede disponer de más de un componente del mismo tipo, por ejemplo, dos cámaras. Este byte contiene la información que permite diferenciar a que componente va dirigido el mensaje. Puede tomar un valor entre 0 y 255.
  - MSG: este byte identifica el mensaje que contiene el payload. Como ya se ha explicado, MAVLink se basa en una librería de mensajes, pues este identificador permite indicar cuál de todos los mensajes es el

que está contenido en el payload. Con esto se puede decodificar el mensaje más fácilmente. Puede tomar un valor entre 0 y 255.

- Payload: contiene la información útil que se quiere transmitir. Debido a que para informar de la longitud solo se dispone de un byte, el tamaño de este campo está acotado entre 0 y 255 bytes.
- Checksum: sirve para comprobar si existen errores en el paquete. Tiene dos para realizar dos comprobaciones, la segunda sirve para evitar conflictos entre diferentes versiones del protocolo. El checksum es el mismo usado en ITU X.25 y SAE AS-4.

4.3 Mensajes

El contenido útil que se quiere transmitir en el payload son los mensajes de la librería que compone MAVLink. Estos mensajes están definidos en XML, un lenguaje marcado que permite almacenar datos de forma legible tanto por los humanos como las máquinas y que es muy fácil de traducir a otros lenguajes.

```
<message id="0" name="HEARTBEAT">
    <description>
    The heartbeat message shows that a system is present and responding.
    The type of the MAV and Autopilot hardware allow the receiving system to treat further messages from this system appropriate
    (e.g. by laying out the user interface based on the autopilot).
    </description>
    <field type="uint8_t" name="type" enum="MAV_TYPE">
    Type of the MAV (quadrotor, helicopter, etc., up to 15 types, defined in MAV_TYPE ENUM)
    </field>
    <field type="uint8_t" name="autopilot" enum="MAV_AUTOPILOT">
    Autopilot type / class. defined in MAV_AUTOPILOT ENUM
    </field>
    <field type="uint8_t" name="base_mode" enum="MAV_MODE_FLAG" display="bitmask">
    System mode bitfield, see MAV_MODE_FLAG ENUM in mavlink/include/mavlink_types.h
    </field>
    <field type="uint32_t" name="custom_mode">
    A bitfield for use for autopilot-specific flags.
    </field>
    <field type="uint8_t" name="system_status" enum="MAV_STATE">
    System status flag, see MAV_STATE ENUM
    </field>
    <field type="uint8_t_mavlink_version" name="mavlink_version">
    MAVLink version, not writable by user, gets added by protocol because of magic data type: uint8_t_mavlink_version
    </field>
</message>
```

Figura 9. Mensaje MAVLink Heartbeat

La figura 9 representa la definición del mensaje *Heartbeat*, que nos va a servir de ejemplo para poder describir como se definen el resto de mensajes MAVLink.

- message></message>: la definición de un nuevo mensaje empieza cuando se encuentra una sentencia <message> y acaba en </message>.

- id: cada mensaje definido en MAVLink tiene que tener un identificador que sirve para poder decodificarlo más fácilmente en el momento de la comunicación.
- name: es un nombre dado al identificador para que sea más entendible por el ser humano. Este nombre no se envía en la comunicación, solo se envía el identificador.
- <description></description>: entre estas dos sentencias se encuentra la descripción de la funcionalidad del mensaje. Esta información tampoco se envía y solo sirve para ayudar a entender su funcionamiento. Aunque es una parte importante, no es obligatorio que el mensaje lo contenga.
- <field></field>: dentro de estas sentencias se define el tipo y el nombre de los parámetros de los que está compuesto el mensaje. Cada uno de los parámetros tiene que ir separado en diferentes sentencias.
- type: define el tipo del dato del parámetro. MAVLink soporta números enteros de tamaño fijo, números en coma flotante, cadena de caracteres y un tipo especial creado por MAVLink.

Como hemos explicado en el apartado anterior, el paquete MAVLink dispone de un byte para poder identificar los mensajes, esto quiere decir que puede diferenciar entre 256 mensajes diferentes. Esto no quiere decir que en el protocolo estén definidos todos estos mensajes, los identificadores comprendidos entre 150 y 230 esta reservados para futuras ampliaciones.

Debido a que MAVLink es un protocolo de uso general para todo tipo de drones, no tiene definidos mensajes para funcionalidades específicas que tengan solo algunos. Pero para no limitar el uso del protocolo, se recomienda diseñar mensajes propios para satisfacer las necesidades requeridas en cada caso. Para ello solo hace falta seguir la estructura antes mencionada y utilizar los identificadores reservados para ampliaciones.

4.4 Definiciones

Como hemos visto, MAVLink se caracteriza por intentar optimizar los recursos al máximo, por lo que utiliza un solo archivo para almacenar las definiciones para todos sus mensajes. Pero en este archivo no solo se definen los mensajes, también es usado para almacenar otro tipo de información.

Esta información sirve para realizar definiciones de estructuras y tipos de fijos de que están contenidos dentro de algunos de los diferentes mensajes. Las figuras 10 y 11 nos pueden servir de ejemplo para entenderlo mejor. En la figura 10 se define el dato *serial_control_flag*, con los valores que puede tomar, y en la figura 11 se puede ver el mensaje que envía dicho dato.

```
<enum name="SERIAL_CONTROL_FLAG">
  <description>SERIAL_CONTROL flags (bitmask)</description>
  <entry value="1" name="SERIAL_CONTROL_FLAG_REPLY">
    <description>Set if this is a reply</description>
  </entry>
  <entry value="2" name="SERIAL_CONTROL_FLAG_RESPOND">
    <description>Set if the sender wants the receiver to send a response as another SERIAL_CONTROL message</description>
  </entry>
  <entry value="4" name="SERIAL_CONTROL_FLAG_EXCLUSIVE">
    <description>Set if access to the serial port should be removed from whatever driver is currently using it,
    giving exclusive access to the SERIAL_CONTROL protocol. The port can be handed back
    by sending a request without this flag set</description>
  </entry>
  <entry value="8" name="SERIAL_CONTROL_FLAG_BLOCKING">
    <description>Block on writes to the serial port</description>
  </entry>
  <entry value="16" name="SERIAL_CONTROL_FLAG_MULTI">
    <description>Send multiple replies until port is drained</description>
  </entry>
</enum>
```

Figura 10. Definición MAVLink Serial_Control_Flag

```
<message id="126" name="SERIAL_CONTROL">
  <description>Control a serial port. This can be used for raw access to an onboard
  serial peripheral such as a GPS or telemetry radio. It is designed to make it possible
  to update the devices firmware via MAVLink messages or change the devices settings.
  A message with zero bytes can be used to change just the baudrate.</description>
  <field type="uint8_t" name="device" enum="SERIAL_CONTROL_DEV">See SERIAL_CONTROL_DEV enum</field>
  <field type="uint8_t" name="flags" enum="SERIAL_CONTROL_FLAG" display="bitmask">See SERIAL_CONTROL_FLAG enum</field>
  <field type="uint16_t" name="timeout" units="ms">Timeout for reply data in milliseconds</field>
  <field type="uint32_t" name="baudrate">Baudrate of transfer. Zero means no change.</field>
  <field type="uint8_t" name="count" units="bytes">how many bytes in this transfer</field>
  <field type="uint8_t[70]" name="data">serial data</field>
</message>
```

Figura 11. Mensaje MAVLink Serial_Control

4.5 Comunicación

Una vez entendido los componentes de los que consta MAVLink, hace falta saber cómo podemos enviar los mensajes para realizar la comunicación. El problema es que, aun siendo un protocolo muy usado, es relativamente nuevo y es difícil encontrar información sobre su funcionamiento, siendo prácticamente su página oficial el único lugar donde podemos encontrarla y, aun así, es bastante escasa. Al final para poder entender como poder enviar correctamente los mensajes ha sido necesario descargarse la implementación que tienen creada e interpretar el código.

MAVLink está originalmente implementado en el lenguaje de programación C, pero debido a la facilidad con la que se puede manejar los archivos XML, es muy fácil traducirlo a otros lenguajes de programación. Para ello, los propios creadores del protocolo han desarrollado una aplicación muy sencilla que, a partir del archivo XML que contiene las definiciones, permite crear una librería en C, C#, Objetive-C, Java, JavaScript, Lua, Python y Swift.

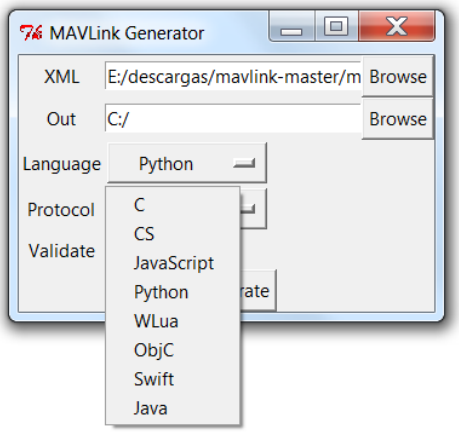

Figura 12. Programa generador de la librería MAVLink

Gracias al estudio de la librería obtenida mediante este programa ha sido posible comprender el proceso necesario para realizar la comunicación con el dron. En este proceso se incluye la creación del mensaje, el encapsulado en un paquete con su correspondiente cabecera, la transformación en una cadena de byte que sea fácil de enviar, y el proceso inverso, que incluye la identificación correcta del mensaje.

Capítulo 5

Diseño

5.1 Introducción

En este capítulo se va a exponer las decisiones de diseño que se han adoptado para obtener una solución final que cumpla con todos los objetivos. Además al final se exponen algunas directrices adicionales que se tienen que tener en cuenta a la hora de realizar la implementación posterior de la aplicación.

5.2 Objetivos

Antes de empezar con la descripción del diseño es recomendable recordar los requisitos a conseguir con la aplicación final.

- Requisitos principales:
  - La aplicación debe ser capaz de controlar el mayor número posible de drones diferentes.

  - La interfaz de control debe de ser idéntica para todos los drones, para que el usuario final que los controle no note la diferencia.

- Requisitos secundarios:
  - Poder manejar más de un dron al mismo tiempo.
  - Poder disponer de un proxy intermedio en la comunicación con los drones, con la posibilidad de colocarlo en otro dispositivo.

5.3 Sistema de comunicación

MAVLink seguramente sea el protocolo de comunicación más usado en el manejo de drones, sobre todo en aquellos fabricados por empresas pequeñas que no se pueden permitir desarrollar uno propio. Si en el desarrollo de la aplicación utilizamos este protocolo seguramente podamos manejar más drones que con cualquier otro, pero habrá otros muchos que no podamos manejar, sobre todo aquellos fabricados por empresas grandes que se puedan permitir desarrollar su propio sistema de comunicación.

En este sentido es conveniente plantarse la posibilidad de utilizar más de un protocolo de comunicación. De esta manera se podría abarcar un mayor número de drones posibles de controlar. Para ello la aplicación debe de poder distinguir con que dron se quiere comunicar y utilizar el protocolo adecuado en cada caso.

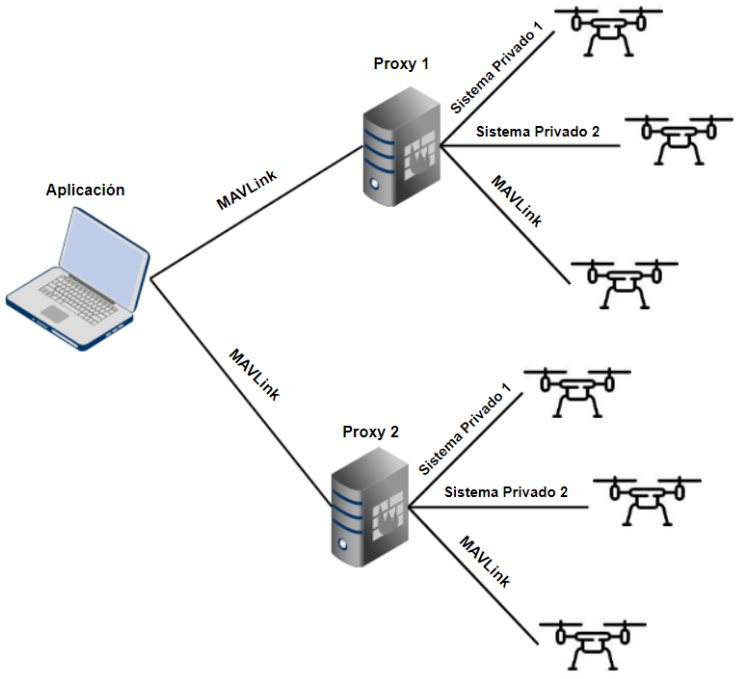

Figura 13. Diagrama del sistema de comunicación

Para cumplir con el objetivo planteado se decidió por un diseño del sistema de comunicación algo más complejo que los anteriores y que responde a lo representado en la figura 13. Este esquema describe una comunicación en la que la aplicación intercambia

la información con un punto intermedio mediante el protocolo MAVLink y desde ahí la comunicación con el dron se realiza mediante el protocolo que sea necesario en cada caso.

Este sistema tiene la ventaja de que nos permite agregar uno o más puntos intermedios de una manera sencilla, permitiéndonos cumplir así con el requisito secundario de agregar un proxy como nos habíamos propuesto. El único condicionante es que para realizar cualquier comunicación se tiene que pasar siempre por un proxy intermedio, pero si este proxy se integra en el mismo dispositivo donde se ejecuta la aplicación, la pérdida de datos en este paso intermedio es nula y el aumento de tiempo que tarda en llegar la información al destinatario final es despreciable.

El resultado final es un sistema que consta de tres partes:

- La aplicación desde donde se decidirá que dron queremos controlar y se realizara el control de dicho dispositivo.
- El proxy, quien actuará como punto intermedio en la comunicación y decidirá que protocolo usar en cada caso.
- Los drones, que tendrán su protocolo de comunicaciones particular que tendrá que integrarse en el proxy.

5.4 Identificación de drones

Una vez decido como se va a realizar la comunicación entre la aplicación y el dron, hay que pensar en cómo realizar la identificación de los drones a la hora de intercambiar información. Además esta identificación cobra más importancia en nuestro caso porque, no solo se debe utilizar para enviar la información al dron con el que nos queremos comunicar, también nos tiene que servir para elegir que protocolo usar en cada caso.

Normalmente para podernos comunicar con otro dispositivo basta con saber su dirección IP y enviarle el mensaje, pero debido a que en el esquema de comunicación elegido donde la información siempre tiene que pasar primero por un proxy intermedio, hay que buscar otra forma de identificar los drones.

En el capítulo 4 hemos visto que en el paquete donde se encapsulan los mensajes MAVLink tiene un parámetro cuya funcionalidad es identificar al dispositivo que lo envía. Este parámetro que nos ofrece MAVLink por sí solo no nos sirve para poder realizar la identificación completa de los drones, ya que es un identificador de origen y no de destino, por lo que a la hora de enviar mensajes tendríamos problemas. Si queremos utilizar este parámetro se tiene que pensar en una manera de solucionar este problema.

Para poder identificar el destino, MAVLink incorpora en algunos de sus mensajes un campo donde se incluye el identificador. Pero este campo no se incluye en todos ellos, esto puede ser debido a que hay mensajes que solo los envían los drones sin necesidad que lo pida la estación de control. Este funcionamiento puede ser útil para drones que implementen MAVLink, pero en nuestro caso también se tienen que poder utilizar otros

protocolos en los que puede ser posible que necesitemos pedir esos datos desde la estación de control.

Con el objetivo de poder utilizar este parámetro para identificar los drones es necesario modificar ligeramente el protocolo MAVLink. Esta modificación se puede hacer de dos maneras diferentes:

- Modificando todos los mensajes para que incluyan un campo que sirva de identificador.
- Utilizando el campo COMP, que ya forma parte del paquete MAVLink, como identificador del dron al que va destinado el mensaje, en vez de su función original de identificar entre diferentes componentes del dron.

Al final se optó por la segunda opción, ya que para cumplir el objetivo de la aplicación no es necesario utilizar en ningún caso el campo COMP. Pero si en un futuro se requiere de la utilización de este campo, se podría recurrir a la primera opción y modificar la aplicación.

Es conveniente aclarar que este protocolo modificado solo es usado entre la aplicación y el proxy intermedio, en la segunda parte de la comunicación, si el dron con el que se quiera intercambiar información utiliza el protocolo MAVLink, se utilizará el protocolo original. Esto se puede hacer gracias a que cuando el mensaje llega al proxy intermedio, la comunicación se hace directamente con los drones, por lo que podemos identificarlos directamente por otros medios como su dirección IP.

5.5 Mensajes MAVLink

Con el esquema de comunicación que hemos elegido, toda la información que se intercambie entre la aplicación y los drones tiene que pasar siempre por una parte donde se utiliza el protocolo MAVLink. Debido a esto, tenemos que elegir los mensajes que nos permitan transmitir la información necesaria para cumplir con todas las funcionalidades que debe tener la aplicación a la hora de manejar el dron.

Las funcionalidades necesarias para la aplicación final que nos hemos planteado son las siguientes:

- Pedir acceso al dron para poder controlarlo.
- Despegar y aterrizar en cualquier momento.
- Tener un control manual del movimiento del dron durante el vuelo.

- Recibir telemetría del dron. Siendo solo necesario conocer la batería del dron, la altura en la que se encuentra con respecto al suelo y saber si está volando o aterrizado.
- Recibir una señal de video en streaming.

MAVLink ya dispone de mensajes definidos para cumplir con estas características sin necesidad de que tengamos que crear nosotros unos propios. Pero de todas formas se han hecho algunos cambios en la manera en que se utilizan los mensajes de control manual y telemetría.

5.5.1 Control manual

Como hemos visto, MAVLink es un protocolo centrado sobre todo en el manejo de drones para piloto automático, donde el control manual no tiene gran relevancia. Seguramente por esto, de los más de cien mensajes que ya tiene definido el protocolo, solo hay uno dedicado a dicho control.

```
<message id="69" name="MANUAL_CONTROL">
    <description>
    This message provides an API for manually controlling the vehicle using standard joystick axes nomenclature,
    along with a joystick-like input device.
    Unused axes can be disabled an buttons are also transmit as boolean values of their
    </description>
    <field type="uint8_t" name="target">The system to be controlled.</field>
    <field type="int16_t" name="x">
    X-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid.
    Generally corresponds to forward(1000)-backward(-1000) movement on a joystick and the pitch of a vehicle.
    </field>
    <field type="int16_t" name="y">
    Y-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid.
    Generally corresponds to left(-1000)-right(1000) movement on a joystick and the roll of a vehicle.
    </field>
    <field type="int16_t" name="z">
    Z-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid.
    Generally corresponds to a separate slider movement with maximum being 1000 and minimum being -1000 on a joystick
    and the thrust of a vehicle. Positive values are positive thrust, negative values are negative thrust.
    </field>
    <field type="int16_t" name="r">
    R-axis, normalized to the range [-1000,1000]. A value of INT16_MAX indicates that this axis is invalid.
    Generally corresponds to a twisting of the joystick, with counter-clockwise being 1000 and clockwise being -1000,
    and the yaw of a vehicle.
    </field>
    <field type="uint16_t" name="buttons">
    A bitfield corresponding to the joystick buttons' current state, 1 for pressed, 0 for released.
    The lowest bit corresponds to Button 1.
    </field>
</message>
```

Figura 14.Mensaje MAVLink Manual_Control

El funcionamiento del mensaje se basa en la manera en que se controlan los vehículos teledirigidos a través de un mando radiocontrol. Para poder manejar los drones desde este tipo de dispositivos se mandan las velocidades de todas las direcciones a la vez, siendo el valor 0 si no se tiene que mover en ese eje de dirección.

Aunque es posible utilizar este mensaje sin modificar para poder controlar el dron, no es óptimo para el uso desde una aplicación donde, en la mayoría de los casos, el interfaz de control manual será un teclado y un ratón. En este punto nos encontramos con dos posibilidades, crear nuestro propio mensaje de control o seguir utilizando el mensaje ya creado. Debido a que el mensaje original es lo suficientemente completo para poder

contralar el dron, se determinó que si modificamos la manera original en la que es usado se podría utilizar perfectamente.

5.5.2 Telemetría

En el protocolo MAVLink el envío de la telemetría la realiza directamente el dron sin necesidad que la estación de control haga ninguna petición. Los datos se envían mediante varios mensajes diferentes en los que pueden ir uno o más valores diferentes de telemetría.

El problema es que en nuestro caso no se va utilizar exclusivamente el protocolo MAVLink y el funcionamiento de otros protocolos puede ser diferente. Por ello se decidió englobar toda la telemetría en un solo mensaje que normalmente se utiliza cuando hay una alta latencia en la comunicación y que incluye todos los valores que necesitamos, aunque menos detallados que en los mensajes específicos. Además, para evitar el caso donde el protocolo requiera la petición de los datos, siempre se tiene que realizar dicha petición al inicio del control del dron.

```
<message id="234" name="HIGH_LATENCY">
    <description>Message appropriate for high latency connections like Iridium</description>
    <field type="uint8_t" name="base_mode" enum="MAV_MODE_FLAG" display="bitmask">
    System mode bitfield, see MAV_MODE_FLAG ENUM in mavlink/include/mavlink_types.h</field>
    <field type="uint32_t" name="custom_mode">A bitfield for use for autopilot-specific flags.</field>
    <field type="uint8_t" name="landed_state" enum="MAV_LANDED_STATE">
    The landed state. Is set to MAV_LANDED_STATE_UNDEFINED if landed state is unknown.</field>
    <field type="int16_t" name="roll" units="cdeg">roll (centidegrees)</field>
    <field type="int16_t" name="pitch" units="cdeg">pitch (centidegrees)</field>
    <field type="uint16_t" name="heading" units="cdeg">heading (centidegrees)</field>
    <field type="int8_t" name="throttle" units="%">throttle (percentage)</field>
    <field type="int16_t" name="heading_sp" units="cdeg">heading setpoint (centidegrees)</field>
    <field type="int32_t" name="latitude" units="degE7">Latitude, expressed as degrees * 1E7</field>
    <field type="int32_t" name="longitude" units="degE7">Longitude, expressed as degrees * 1E7</field>
    <field type="int16_t" name="altitude_amsl" units="m">Altitude above mean sea level (meters)</field>
    <field type="int16_t" name="altitude_sp" units="m">
    Altitude setpoint relative to the home position (meters)</field>
    <field type="uint8_t" name="airspeed" units="m/s">airspeed (m/s)</field>
    <field type="uint8_t" name="airspeed_sp" units="m/s">airspeed setpoint (m/s)</field>
    <field type="uint8_t" name="groundspeed" units="m/s">groundspeed (m/s)</field>
    <field type="int8_t" name="climb_rate" units="m/s">climb rate (m/s)</field>
    <field type="uint8_t" name="gps_nsat">Number of satellites visible. If unknown, set to 255</field>
    <field type="uint8_t" name="gps_fix_type" enum="GPS_FIX_TYPE">See the GPS_FIX_TYPE enum.</field>
    <field type="uint8_t" name="battery_remaining" units="%">Remaining battery (percentage)</field>
    <field type="int8_t" name="temperature" units="degC">Autopilot temperature (degrees C)</field>
    <field type="int8_t" name="temperature_air" units="degC">
    Air temperature (degrees C) from airspeed sensor</field>
    <field type="uint8_t" name="failsafe">
    failsafe (each bit represents a failsafe where 0=ok, 1=failsafe active
    (bit0:RC, bit1:batt, bit2:GPS, bit3:GCS, bit4:fence)</field>
    <field type="uint8_t" name="wp_num">current waypoint number</field>
    <field type="uint16_t" name="wp_distance" units="m">distance to target (meters)</field>
</message>
```

Figura 15. Mensaje MAVLink High_Latency

5.5.3 Diagrama de mensajes

En el diagrama de la figura 16 se muestra el intercambio de mensajes que se tienen que realizar entre la aplicación y el proxy en una sesión normal en la que se controle un dron. La comunicación entre el proxy y el dron se tendrá que hacer según los estándares del protocolo necesario en cada caso.

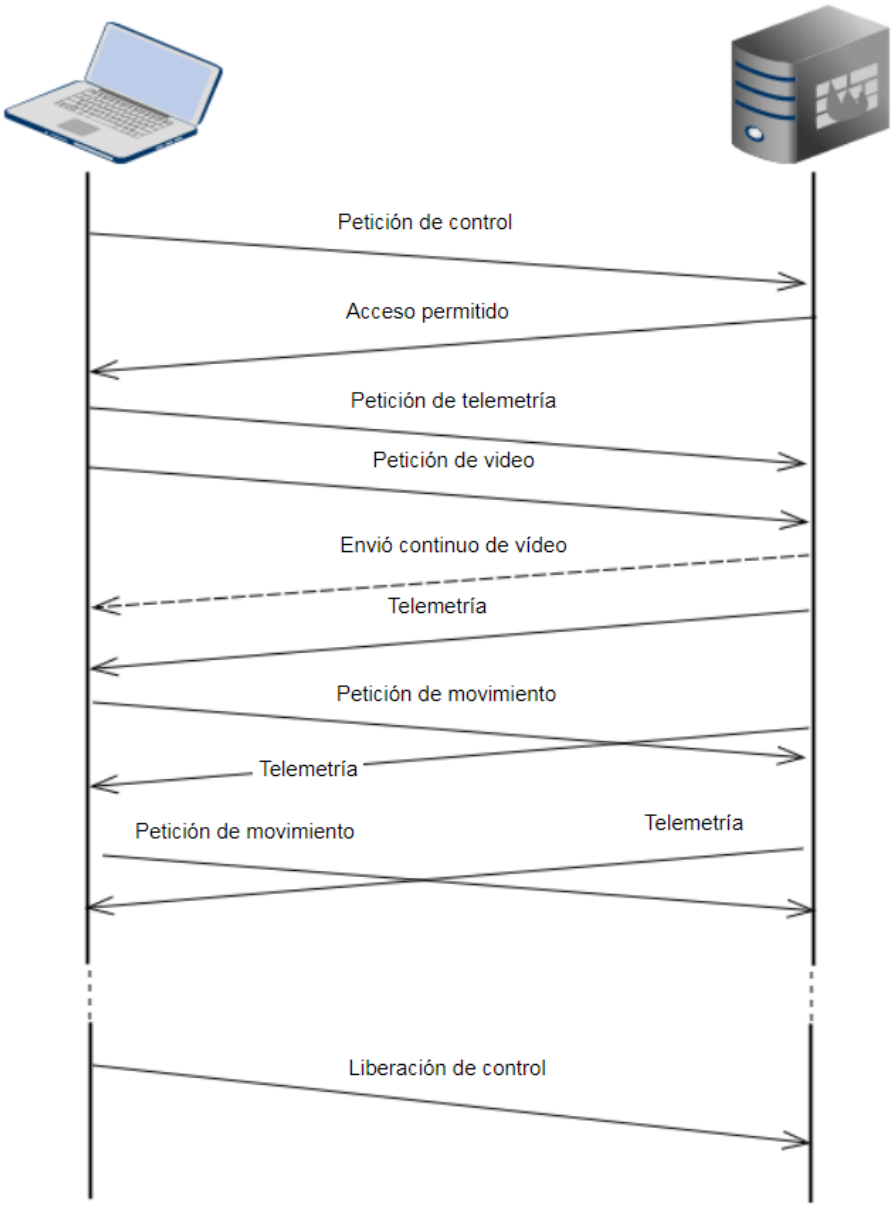

Figura 16. Diagrama de mensajes

Capítulo 6

Implementación

6.1 Introducción

En este capítulo se va a exponer las decisiones que se han tenido que tomar para poder realizar la programación de la aplicación siguiendo el diseño marcado en el apartado anterior.

6.2 Lenguaje de programación

Elegir el lenguaje de programación usado a la hora de crear una aplicación es muy importante, ya que una buena elección puede facilitar el proceso de programación y la implementación en diferentes sistemas operativos y dispositivos.

En el capítulo 4 hemos visto que los creadores de MAVLink han desarrollado un programa que facilita la creación de librerías en los siguientes lenguajes de programación: C, C#, Objetive-C, Java, JavaScript, Lua, Python y Swift. Para poder elegir entre todos ellos tenemos que saber en qué se diferencian.

- C**:** Es un lenguaje orientado a la implementación de sistemas operativos. Es un lenguaje de tipos de datos estáticos, débilmente tipificado, de medio nivel, ya que dispone de estructuras de lenguajes de alto nivel, pero también cuenta con otras que permiten el control de bajo nivel.
- Objetive-C**:** Es un lenguaje orientado a objetos creado como un superconjunto del lenguaje C.
- C#: Otro lenguaje orientado a objetos basado en C. La principal diferencia con respecto a Objetive-C es que para su creación no se utilizó solo C como referencia, sino que influyeron más lenguajes.
- Java: Es un lenguaje orientado a objetos y diseñado específicamente para que sea fácil de implementar en diferentes ambientes. Para conseguir esto el código Java no se ejecuta directamente, sino que lo tiene que hacer sobre una máquina virtual Java (JVM) que tiene que estar instalada previamente en el dispositivo. De esta manera una aplicación programada en este lenguaje no tiene por qué compilarse para cada arquitectura diferente.
- JavaScript: Es un lenguaje de programación interpretado, débilmente tipado y dinámico. Su objetivo original es poder ser interpretado fácilmente por los navegadores web para crear páginas web dinámicas.
- Lua: Es un lenguaje imperativo, estructurado y bastante ligero que fue diseñado como un lenguaje interpretado con una semántica extensible.
- Python: Es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.
- Swift: Es un lenguaje de programación creado por Apple enfocado en el desarrollo de aplicaciones para MacOS e iOS.

De todos estos lenguajes, Java y C y sus derivados, son los más utilizados para realizar aplicaciones multiplataforma. A la hora de elegir entre uno de estos lenguajes se decidió utilizar Java debido a la facilidad con la que se puede implementar en más de un sistema operativo, con lo que podemos conseguir que la aplicación sea utilizada por más gente con menos dificultades.

6.3 Aplicación

En esta parte del programa se encuentra la interfaz gráfica con la que va interactuar el usuario. Desde ella se debe poder mandar los comandos de control al dron y recibir tanto la telemetría como el video. Además, hay que tener en cuenta que todo este proceso se tiene que poder realizar en paralelo si se realizan vuelos simultáneos de más de un dron.

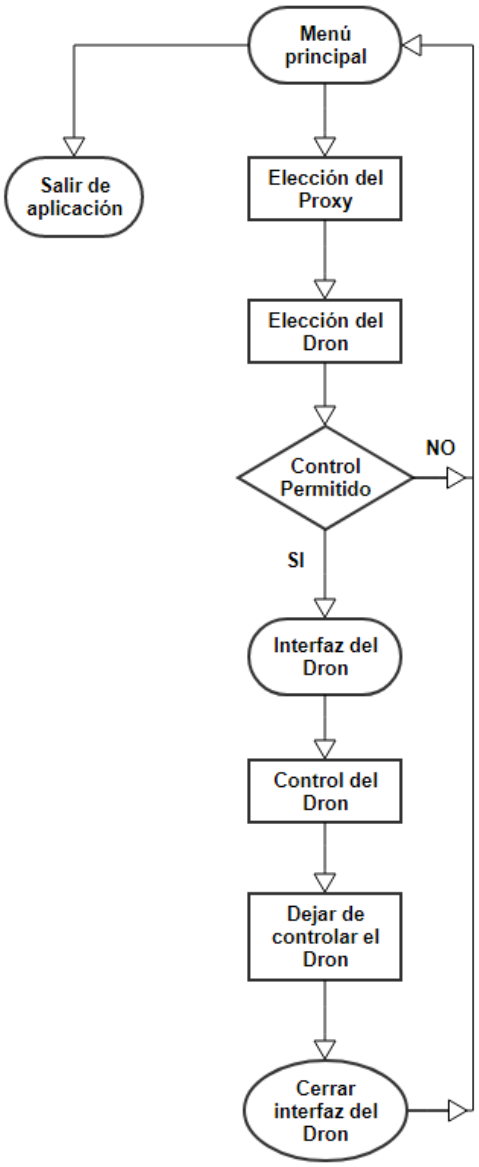

Figura 17. Diagrama de flujo de la aplicación

La figura 17 representa el diagrama de flujo de la aplicación, donde se muestra el funcionamiento normal cuando se realiza el control de un dron. Pero la aplicación debe poder realizar el control de más de un dispositivo a la vez, por lo que hay que modificar este esquema para poder realizar esta funcionalidad.

Además, hay que tener en cuenta que durante el funcionamiento del programa, el menú principal siempre tiene que estar abierto para poder seleccionar otro dron o para salir de la misma, por lo que cada vez que queremos controlar un dron, su panel de control tiene que salir en una ventana o pestaña diferente.

6.3.1 Manejar varios drones a la vez

A la hora de programar la aplicación para que se pueda controlar más de un dron de manera independiente, hay que tener en cuenta que cada uno de los drones tiene que estar enviando y recibiendo información de manera constante, este funcionamiento puede generar que la información relativa un dron sea recibida por otro diferente. Este es un problema muy grande y hay que crear la aplicación de tal manera que no exista ninguna posibilidad de que pase.

Una de las posibles soluciones para este problema es utilizar puertos diferentes por cada uno de los drones. De esta manera se establece una comunicación independiente con cada uno de ellos.

Pero solo usando diferentes puertos no basta para asegurarnos la independencia del control entre diferentes drones. Para asegurarnos que cada dron se manejase de manera totalmente independiente, sin que existiese ninguna posibilidad de interferencia entre ellos, se decidió separar la ejecución de cada dron en un hilo separado. Con ello conseguimos que cada dron utilice sus propios recursos del sistema y que si en algún momento existe un problema con uno de ello, no interfiera con los demás.

De esta manera el diagrama de flujo que se muestra en la figura 17 seguiría siendo válido, solo que hay que tener en cuenta que, durante el funcionamiento del programa, el menú principal siempre tiene que estar abierto para poder seleccionar otro dron. De esta manera, cuando se seleccione el manejo de un dron, se crea un nuevo hilo que abrirá su propia interfaz gráfica en otra ventana.

6.3.2 Agregar drones

La tarea de agregar nuevos drones no está pensada para que la realice el usuario final, ya que no es tan simple como modificar la interfaz agregando otro dispositivo, además se tendrán que implementar en el proxy el protocolo del nuevo dron. Por esta razón no se ha buscado una solución en la que el usuario final agregue directamente los drones desde la interfaz gráfica. De todas formas, aunque la modificación la deba hacer una persona con conocimientos de programación, es recomendable crear una aplicación en la que dichas modificaciones sean mininas y fáciles de realizar.

Como hemos visto en el apartado anterior, para poder diferenciar que dron se está utilizando se utilizan dos cosas: hilos y puertos diferentes. Al haber decidido utilizar hilos, el código que utilizan todos los drones es el mismo, pero cada uno de ellos tiene que poder realizar la comunicación con diferentes puertos. Para poder solucionar este problema, se decidió incluir en el código común para todos los hilos estructuras *switch* en las partes que se hace referencia a los puertos. En esta estructura colocamos los puertos de los drones que estén configurados y en el momento de la ejecución cada hilo sabrá cuál es el puerto que tiene que elegir. De esta manera agregar drones es tan fácil como añadir un nuevo caso en los *switch* necesarios.

```
int puerto;
switch(drone){
    case 1:
        puerto=2051;
        break;
    case 2:
        puerto=2052;
        break;
}
```

Figura 18. Ejemplo de *switch* en el código

6.3.3 Interfaz grafica

El diseño de la interfaz gráfica es uno de los procesos más importantes a la hora de crear una aplicación, ya que es la parte que el usuario final va ver y manejar, y crear un diseño que no sea fácil de comprender, dificulte su uso o no tenga un aspecto visual atractivo, puede hacer que una buena aplicación fracase.

Con el fin diseñar una interfaz amigable y funcional es necesario definir cuáles son los requisitos que queremos que cumpla:

- Debido a que no sabemos quién va ser el usuario final, es necesario que sea simple para que pueda ser comprendida fácilmente sin necesidad de ninguna explicación.
- Debe disponer de un menú principal desde el cual se pueda seleccionar el dron que queremos manejar.
- En el menú principal se debe mostrar de forma clara a través de que proxy se va comunicar con los drones.
- La interfaz gráfica debe ser idéntica para todos los drones.
- La interfaz desde la que se controla los drones debe mostrar claramente tres cosas:
    - Un panel desde el cual podamos controlar al dron.
    - Los datos de la telemetría.
    - La reproducción del video obtenido del dron.
- El panel de control del dron tiene que diseñarse de tal manera que sea fácilmente manejable con un ratón y un teclado.

6.3.3.1 Menú principal

Como hemos especificado en los requisitos, la aplicación debe disponer de un menú principal desde el cual se seleccione el dron o los drones que queremos manejar. Hay que pensar que la mayoría de la gente no dispone de un elevado número de drones diferentes y que, en la mayoría de los casos, este número se reducirá a dos o tres modelos como máximo. Teniendo esto en cuenta, en vez de realizar un menú donde la selección de los

drones se realice mediante grandes listas y submenús, donde es más fácil que el usuario se pierda, se diseñó un menú donde la selección se hace mediante botones.

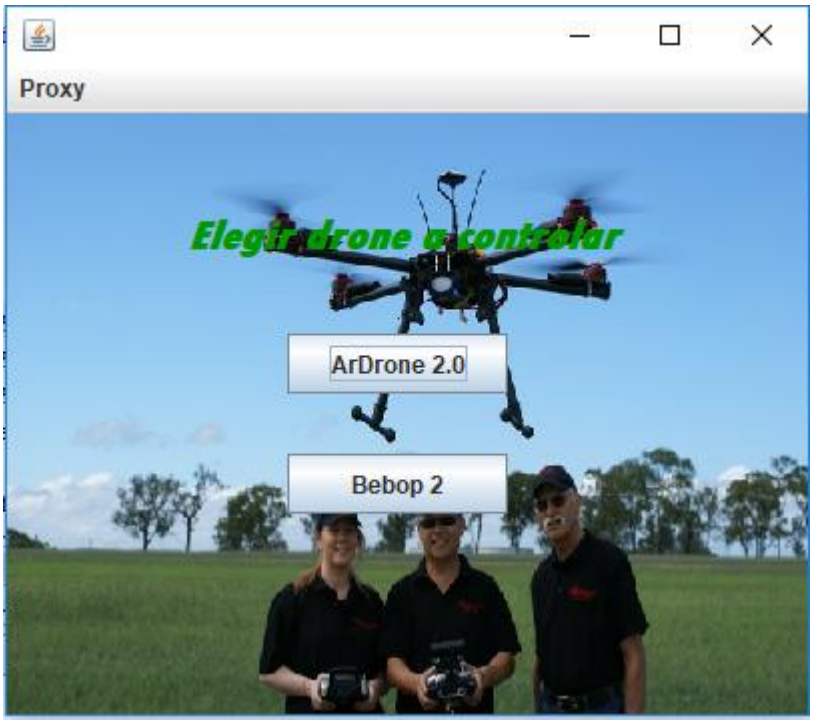

Figura 19.Interfaz gráfica, menú principal

En la figura 19 se puede ver el sencillo menú principal que permite elegir fácilmente que dron queremos controlar, pero además de ello en la escuadra superior derecha se observa un menú donde se puede seleccionar el proxy.

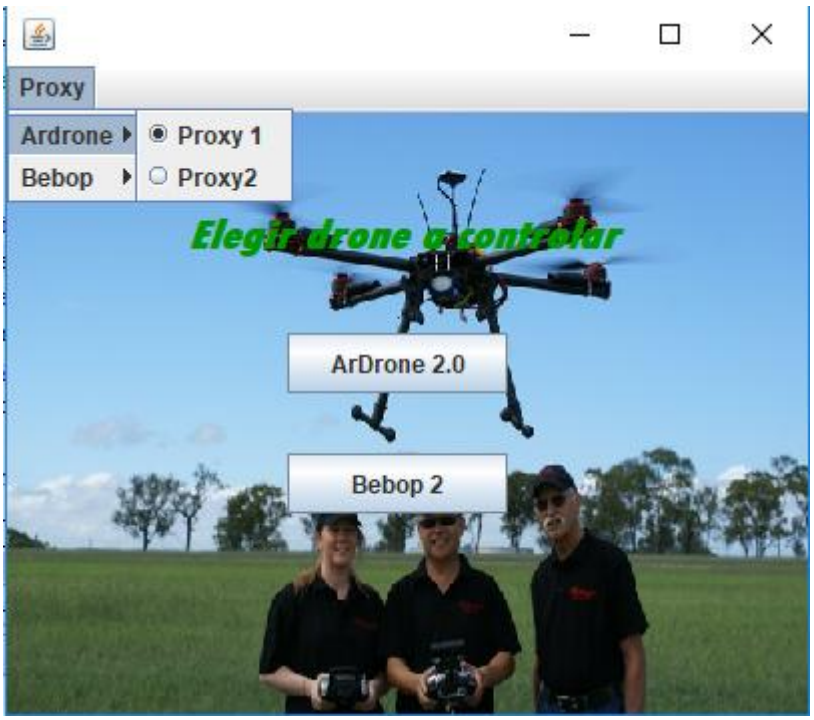

Figura 20. Interfaz gráfica, selección de proxy

Como se observa en la figura 20, por cada dron se puede elegir el proxy que deseamos utilizar. En este caso se prefirió utilizar un menú para seleccionar el proxy ya que normalmente siempre se utilizará la misma configuración y si cada vez que queremos controlar un dron debemos elegir un proxy, lo que estaríamos haciendo es añadir un paso más innecesario.

6.3.3.2 Mensajes

Cuando queremos controlar un dron y apretamos el botón que lo selecciona, antes de que se abra el interfaz que nos permita controlarlo, es posible que se muestren mensajes informativos.

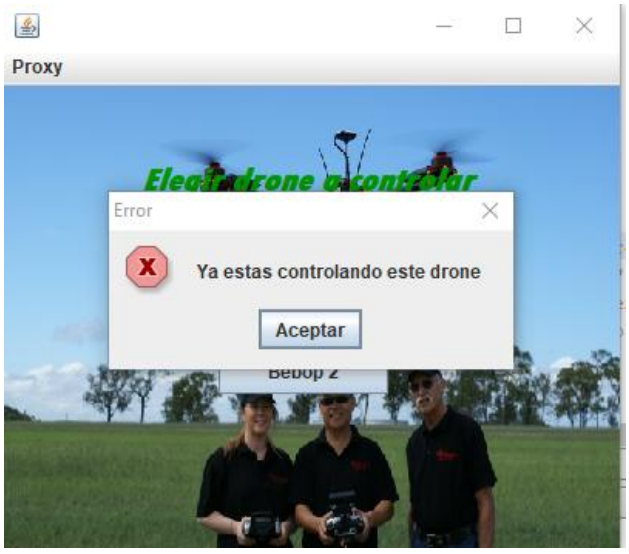

Figura 21. Interfaz gráfica, mensaje 1

El mensaje de la figura 21 se muestra cuando ya estas controlando dicho dron, por lo que no se puede crear otra interfaz de control que interfiera con la que ya está en funcionamiento.

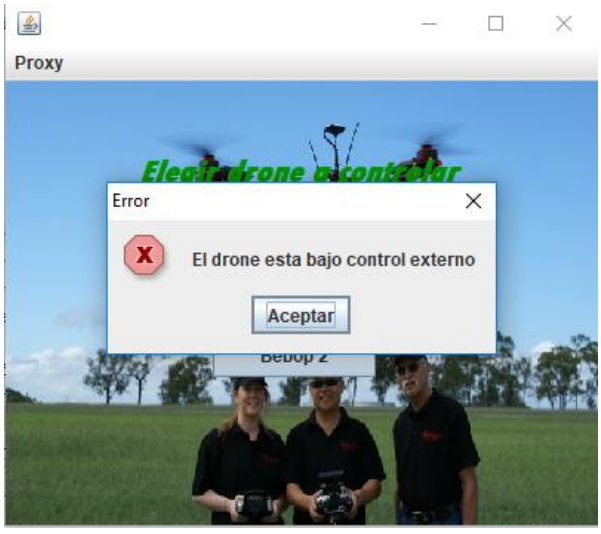

Figura 22. Interfaz gráfica, mensaje 2

El mensaje de la figura 22 se muestra cuando el dron responde a la petición de control denegando el acceso por que ya está bajo el control de otra aplicación.

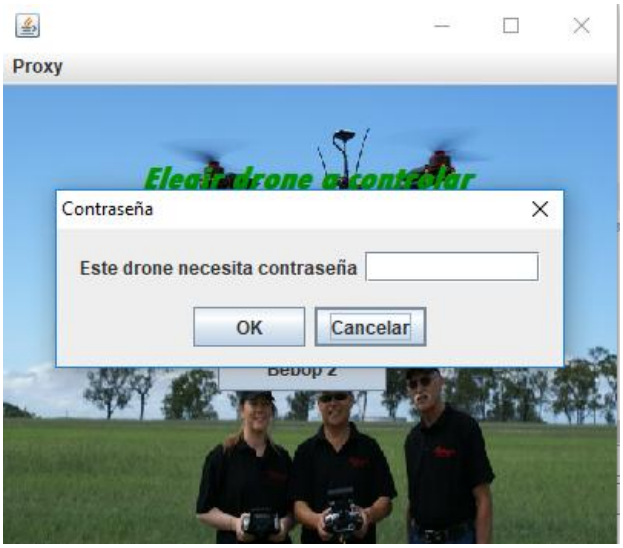

Figura 23. Interfaz gráfica, mensaje 3

El mensaje de la figura 23 se muestra cuando el dron requiere de una contraseña para poder controlarlo.

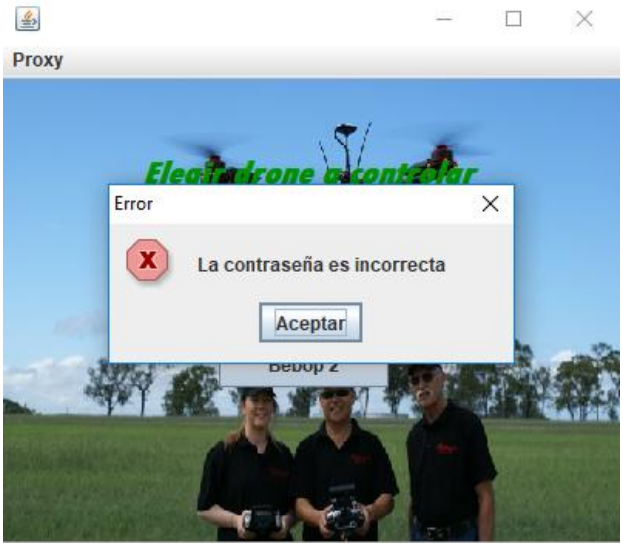

Figura 24. Interfaz gráfica, mensaje 4

Este último mensaje mostrado en la figura 24, surge cuando la contraseña introducida no es válida.

6.3.3.3 Interfaz de control

En esta parte de la interfaz es donde se encuentra el panel de control del dron, y para diseñarlo hace falta pensar en la mejor manera de controlar al dron mediante un ratón y un teclado.

El control mediante el teclado, ya sea introduciendo valores o directamente apretando teclas que tengan un valor fijo, puede ser un método muy eficiente si se tiene practica en su manejo, pero el objetivo de esta aplicación es que la pueda usar cualquier usuario, por lo que es un método que queda descartado. El otro método que nos queda es el uso del ratón, y la manera más precisa de manejar al dron con este dispositivo es creando un panel de control donde las distintas velocidades se muestren en botones diferentes.

Figura 25. Interfaz gráfica, panel de control

En la figura 25 se muestra la interfaz final creada para manejar el dron. En ella se distinguen tres partes diferentes:

- La parte superior muestra la reproducción del video que se obtiene en streaming desde el dron.

- En la parte intermedia hay una franja en la que se muestra el valor actualizado de la telemetría.
- En la parte inferior se encuentra el panel de control que muestra el conjunto de botones desde el cual se manejará el dron. En esta parte se incluye un fondo con la imagen del dron que se está manejando en este momento, lo que permite que se distinga mejor cuando se está manejando más de un dispositivo a la vez.

6.3.3.3.1 Reproductor de video

La obtención y reproducción del video es una de las partes del proyecto que ha llevado más tiempo realizar. El problema que encontramos al realizar esta tarea se debe principalmente a los formatos de codificación de video que normalmente utilizan los drones a la hora de transmitir el video en vivo. Estos formatos son relativamente nuevos y permiten una compresión mayor del tamaño del video, pero la mayoría de los frameworks para el tratamiento de video que existen en Java son bastante antiguos y no son capaces de comprenderlos.

Para encontrar un método con el que se pudiese reproducir el video se realizó una extensa búsqueda. Al no encontrar una solución convencional, se desarrolló un sistema más complejo en el que el dron tenía que tomar fotografías de manera continua y reproducirlas a una velocidad que se pudiese simular un video. Este sistema no funcionó porque la velocidad de escritura del disco duro del ordenador no permitía una reproducción aceptable.

Después de realizar otra intensa búsqueda se encontró un framework que permitía incrustar el reproductor VLC dentro de la aplicación. Utilizar este framework tiene la desventaja que tenemos que tener instalado otro programa externo para el funcionamiento de la aplicación. Pero esta desventaja a la vez es una ventaja, gracias a que es un reproductor que está continuamente actualizándose y si en un futuro los drones empiezan a utilizar un nuevo formato, no hace falta realizar modificaciones en la aplicación, solo es necesario actualizar el programa externo.

Este framework reproduce el video desde un URL (Uniform Resource Locator), por lo que para pueden existir dos casos a la hora de obtener el video que emiten los drones:

- Que el dron pueda emitir directamente a un URL. En este caso solo habría que configurar el framework para obtener el video desde ese recurso.
- Que el dron no pueda emitir por sí mismo a un URL. En este caso tendríamos que realizar el tratamiento del video previamente y crear un propio URL con el cual el framework pueda trabajar.

Independientemente de cuál sea el caso, cuando la comunicación con el dron se realice a través de un proxy externo que no esté instalado en el mismo dispositivo que la aplicación, todo el flujo de datos del video se tiene que enviar a través de éste, por lo que se tendrán que agregar a dicho proxy el código necesario para realizar este reenvió.

6.4 Proxy

La proxy tiene dos funciones, redirigir los mensajes como punto intermedio en la comunicación entre la aplicación y los drones, y elegir que protocolo de comunicación elegir en cada caso.

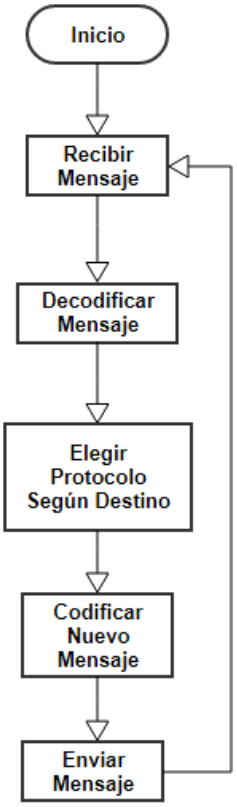

Figura 26. Diagrama de flujo del proxy

Como se observa en el diagrama de flujo representado en la figura 26, el proxy actúa como un servidor que está esperando continuamente mensajes para poder enviarlos al destinatario indicado. La peculiaridad es que en vez de realizar el envío directamente, tiene que elegir que protocolo usar con el destinatario y crear un mensaje adecuado a ello.

6.4.1 Agregar drones

La tarea de agregar nuevos drones no está pensada para que la realice el usuario final, ya que no es tan simple como modificar la interfaz agregando otro dispositivo, además se tendrán

Como hemos visto, en el proxy tiene que estar implementado los protocolos necesarios para poder comunicarnos con todos los drones configurados en la aplicación, por lo que en un principio podríamos pensar que cuando agreguemos un nuevo dron es necesario implementar su protocolo en el proxy, pero en realidad existes tres casos diferentes a la hora de realizar esta tarea:

1. El dron utiliza un protocolo que ya está implementado.
2. El dron utiliza un nuevo protocolo que se puede programar en el mismo lenguaje en el que está hecha la aplicación.
3. El dron utiliza un nuevo protocolo que no se puede programar en el lenguaje en el que está hecha la aplicación.

Las modificaciones que se tendrían que hacer en cada caso serían respectivamente las siguientes:

1. Solo sería necesario agregar una nueva dirección de envió y asociar este nuevo dron al protocolo que ya existe.
2. Además de lo anterior, habría que programar un nuevo protocolo.
3. Este es un caso más complejo que los anteriores. Como no se puede realizar la implementación directamente en el mismo proxy, hay que programarlo fuera de él.

Para poder realizar la comunicación en el caso 3 hace falta crear una comunicación entre el proxy y el programa donde se ejecuta el nuevo protocolo. Esta comunicación requiere de dos cosas, la primera es crear los programas donde se ejecutan los protocolos externos como servidores independientes, y la segunda es crear unos mensajes propios que permitan realizar esta comunicación.

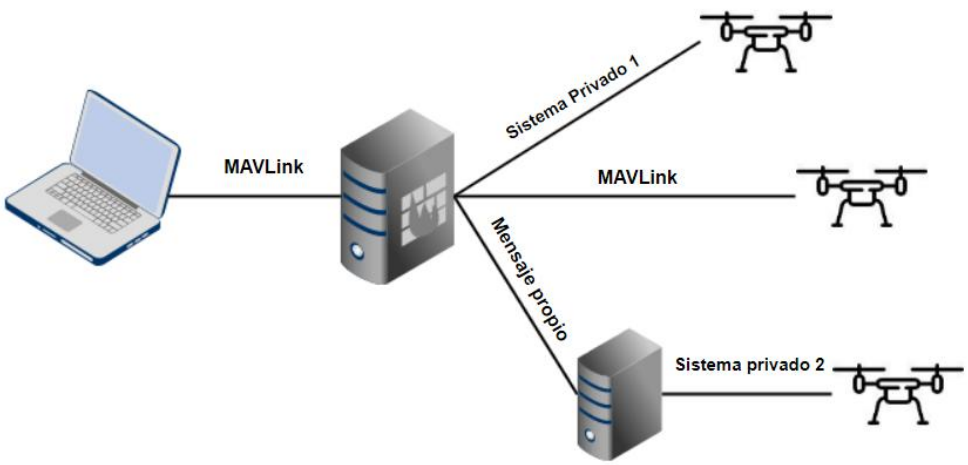

Figura 27. Diagrama del sistema de comunicación con protocolo externo

6.4.1.1 Mensajes propios

La función de los mensajes que comunican el proxy con el programa donde se ejecuta el protocolo exterior, es transmitir la misma información que los mensajes MAVLink transmiten en la primera parte de la comunicación. Debido a esto podríamos

pensar en utilizar directamente el mismo mensaje, pero existe la posibilidad de que en el lenguaje en el que este programado el protocolo externo sea difícil implementar MAVLink. Para evitar futuros fallos, se decidió crear unos mensajes más sencillos a partir de la estructura del paquete MAVLink.

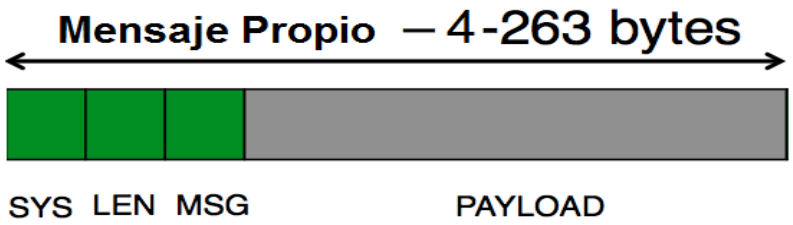

Figura 28.Mensaje propio

En la figura 28 se muestra la estructura que tiene los nuevos mensajes, como se puede ver difiere de la estructura original vista en el capítulo 4, habiendo eliminado en este caso varios bytes. Las razones para eliminarlos son las siguientes:

- STX: al no ser un mensaje MAVLink no es necesario un byte que lo identifique como tal
- SEQ: en esta aplicación los mensajes que se van a utilizar son cortos y no es necesario separarlos en más de una parte, por lo que no es necesario un número de secuencia.
- COMP: para la finalidad de esta aplicación no es necesario diferenciar entre los componentes de un dron.
- Checksum: los bytes de comprobación de errores no son necesarios porque al ser mensajes que solo se van a transferir dentro del mismo sistema, la posibilidad de fallos es mínima.

6.5 Diagrama de clases

En la figura 29 se muestra el diagrama de clases de la aplicación y el proxy. Realmente son dos bloques separados, pero ambos tiene que disponer de la clase Destinos, ya que es una clase destina a contener los atributos constantes que hacen referencia a las IPs, números de puerto y otros valores necesarios para la comunicación.

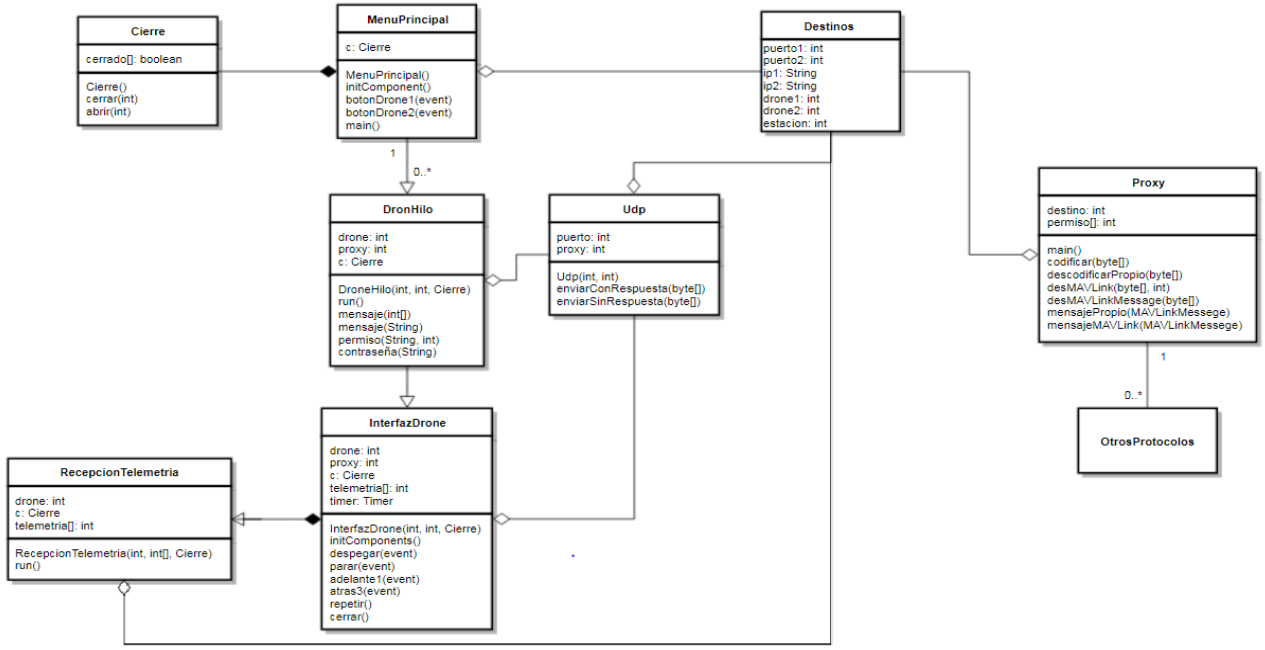

Figura 29. Diagrama de clases

Capítulo 7

Validación

7.1 Introducción

En este capítulo se va a describir todo el escenario de pruebas que se ha tenido que realizar para comprobar que la solución final cumple con todos los requisitos que nos habíamos planteado y que no existe ningún fallo en su funcionamiento.

7.2 Planteamiento

Para que podamos asegurar que la aplicación cumple con todos los requisitos que nos hemos planteado es necesario que se realicen unas pruebas que verifiquen su funcionamiento.

Las pruebas que van a evaluar el correcto funcionamiento de la aplicación son las siguientes:

1. Control de cada uno de los drones.
    - En esta prueba se tendrá que comprobar que se tiene un manejo completo de cada uno de los drones implementados.
2. Control simultaneo de más de un dron.
    - En esta prueba se comprobará que si tenemos abierto el panel de control de más de un dron, podemos manejar cada uno de ellos de manera independiente.
3. Implementar un proxy en un dispositivo diferente de donde se ejecuta la aplicación.
    - Esta prueba debe comprobar que al implementar un proxy en un dispositivo externo, el control de drones funciona de la misma manera que si el proxy estuviera implementado en el mismo dispositivo donde se ejecuta la aplicación.
    - En esta prueba también se debe comprobar que se puede trabajar con más de un proxy a la vez.

7.3 Elección de dispositivos

Para poder ejecutar las pruebas es necesario contar con los dispositivos que nos permitan realizarlas. Es necesario disponer de un ordenador donde se ejecute la aplicación, dos drones y otro ordenador donde poder instalar el proxy externo.

7.3.1 Ordenador principal

El ordenador principal necesario para ejecutar la aplicación no tiene por qué ser muy potente y solo requiere que disponga de un adaptador wifi con el que podamos comunicarnos con los drones. Estas características solo excluyen a los ordenadores fijos que no dispongan de adaptador wifi, por lo que cualquier portátil del mercado es válido. Para realizar las pruebas se usó el mismo portátil personal que se usó para el desarrollo de la aplicación.

Portátil principal	
Modelo	Lenovo G500s
Sistema Operativo	Microsoft Windows 10
Procesador	Intel Core I7 3612QM
Memoria RAM	16 Gb
Disco Duro	500 Gb

Tabla 1. Portátil principal

7.3.2 Drones

La razón por la que se realiza este proyecto es principalmente por el gran desarrollo que está teniendo en los últimos años el sector de los drones. Gracias a esto podemos encontrar en el mercado una gran cantidad de dispositivos en un abanico de precios bastante grande, dependiendo de su calidad y funcionalidades.

- DJI Phantom 4: en cuestión de drones la marca DJI es la reina del mercado, fabricando modelos tanto para los consumidores más exigentes como para profesionales. El modelo Phantom 4 es una de sus estrellas, aunque según la clasificación interna de la marca, es un modelo no profesional pero cuenta con las características para serlo: es un dron muy estable, con un rango de control de hasta 5 kilómetros, una autonomía de 28 minutos, velocidad de hasta 72 km/h, una cámara que puede grabar en resolución 4K y hace fotografías de 12,5 megapíxeles, y cuenta con sensores y software que evitan colisionar con obstáculos durante el vuelo.

- DJI Phanton 3: es el hermano pequeño del modelo anterior, pero de todas formas tiene muchas características que le hacen un gran dron: no es tan estable, tiene un rango de control de 500 metros, una autonomía de 25 minutos, puede volar con una velocidad de hasta 16 km/h y su cámara puede grabar en una resolución máxima de 2.7K y hacer fotografías de 12 megapíxeles.

- Parrot Bebop 2: el último modelo de la marca Parrot es un dron de tamaño reducido pero muy potente y estable. Tiene un rango de control de 300 metros, una autonomía de 25 minutos, puede volar con una velocidad de 65 km/h y cuenta con una cámara que graba en Full HD y hace fotografías con 14 megapíxeles de resolución.

- Parrot Ar Drone 2.0: es un modelo antiguo de la compañía Parrot, pero en su momento se vendió muy bien y hay una gran comunidad y muchos proyectos hechos con él. Tiene un rango de control de solo 50 metros, una autonomía de 12 minutos, vuela con una velocidad máxima de 40 km/h y su cámara graba en HD Ready y hacer fotografías.

Figura 30. Parrot Bebop 2 (Fuente: global.parrot.com)

Aunque existen muchas marcas en el mercado, para realizar este proyecto nos hemos centrado en estas dos porque sus productos son fiables, fáciles de conseguir y hay una gran comunidad de gente entorno a ellos. En la tabla 2 se resumen más claramente las diferencias entre los drones que se podrían utilizar.

	Phantom 4	Phantom 3	Bebop 2	Ar Drone 2.0
Rango de control	5 km	500 m	300 m	60 m
Velocidad	72 km/h	16 km/h	65 km/h	40 km/h
Autonomía	28'	25'	25'	12'
Video	4K (4096x2160)	2.7K (2704x1520)	Full HD (1920x1080)	HD Ready (1080x720)
Fotografía	12.5 Mp	12 Mp	14 Mp	HD
Precio	1200€	498€	480€	210€

Tabla 2. Selección de drones

Como se aprecia en la tabla los drones de DJI tienen mejores características pero son más caros. Para realizar las pruebas de este proyecto no son necesarias muchas funcionalidades, se necesitan drones estables que respondan bien a los comandos y que se puedan manejar dentro de una habitación cerrada, ya que es el lugar donde se realizarían las pruebas. El primer dron que se eligió fue el Parrot Ar Drone 2.0 porque la universidad ya disponía de uno de ellos. El DJI Phantom 4 se descartó por su elevado precio. Entre el DJI Phantom y el Parrot Bebop 2, se eligió este último porque era muy estable y pequeño, lo que permitía un fácil manejo dentro de una habitación. Además se comprobó que las librerías utilizadas para su manejo eran diferentes.

7.3.3 Proxy externo

El ordenador necesario para poder instalar el proxy externo puede ser incluso menos potente que el necesario para instalar la aplicación, solo se necesita un ordenador que sirva de servidor sencillo. Con estas características la mejor opción es elegir uno de los nuevos mini-ordenadores que hay actualmente en el mercado, ya que su coste es reducido y se pueden transportar fácilmente.

En este caso no hubo muchas dudas a la hora de escoger el dispositivo, eligiendo el mini-ordenador Raspberry Pi 3 modelo B. Es un ordenador muy básico, siendo prácticamente una pequeña placa base que con el procesador integrado, memoria RAM, un módulo wireless y puertos para poder conectarla a otros dispositivos.

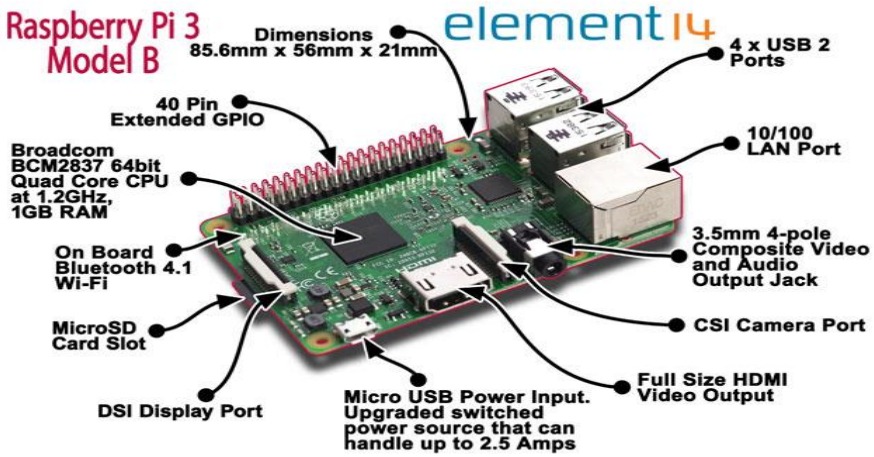

Figura 31. Raspberry Pi 3 modelo B (Fuente: www.element14.com)

Como se puede ver en la figura31 es un dispositivo tan pequeño que se puede montar en los propios drones, con esto podemos conseguir, entre otras cosas, disponer de un "repetidor" de cobertura móvil o, si se le añaden módulos al Raspberry Pi, podemos dotar a los drones de funcionalidades que no traen de fábrica, como una segunda cámara o sensores meteorológicos.

Raspberry Pi 3 Modelo B
Quad Core 1.2GHz Broadcom BCM2837 64bit
1GB RAM
BCM43438 wireless LAN and Bluetooth
1 x Ethernet
1 x 40 GPIO pin
1 X HDMI 1.4
4 x USB 2.0
1 x Combo audio/micrófono
1 x Interfaz de cámara (CSI)
1 X Interfaz de Pantalla (DSI)
1 x Micro SD

Tabla 3. Raspberry Pi 3 modelo B

7.4 Agregar drones

Antes de poder empezar a realizar las pruebas es necesario implementar, tanto en la aplicación como en el proxy, los dos drones que hemos elegido usar para éstas.

Al haber elegido dos drones con gran distribución en el mercado existe una gran comunidad entorno a ellos. Gracias a esto se puedo encontrar dos librerías diferentes a las ofrecidas por el fabricante, pero más fáciles de comprender que las originales. Aunque las dos librerías son independientes, para desarrollar la que controla el Parrot Bebop 2 se basaron en la librería usada para manejar el Parrot Ar Drone 2.0, por lo que una vez entendido el funcionamiento de una es relativamente sencillo comprender el funcionamiento de la otra.

Estas librerías no se pudieron implementar en el código del proxy, ya que el lenguaje en la que estaban programadas es JavaScript, diferente al lenguaje Java usado en el proxy. Para poder realizar dicha implementación se tuvo que crear un programa aparte por cada uno de los drones y realizar la comunicación con ellos mediante mensajes propios, como se explica en el capítulo 6.

7.5 Conectar dispositivos

Para realizar las pruebas es necesario que los dispositivos que están implicados en sus desarrollos estén conectados a una misma red para que puedan comunicarse entre ellos.

Normalmente los drones en un funcionamiento común actúan como un propio router que crea su propia red y asignas las direcciones IPs. Para realizar las pruebas hay que configurar los router para que tengan con este comportamiento, sino que puedan actuar como un dispositivo común que recibe una dirección IP. Realizar este proceso es relativamente fácil en los dos drones que hemos seleccionado, solo hace falta conectarnos vía Telnet al dispositivo y cambiar la configuración como un ordenador Linux normal.

Al final, aunque se compró que la Raspberry Pi 3 modelo B podría actuar como router, se usó el dron Parrot Bebop 2 como punto de acceso para poder comprobar las capacidades del aparato.

7.6 Pruebas

En los siguientes apartados se van a presentar en unas tablas donde se evalúan de 1 a 5 los resultados de las pruebas, siendo 1 el fallo completo de la aplicación en el requisito en evaluación y un 5 el funcionamiento perfecto en el mismo.

Estos resultados corresponden a las pruebas finales, tras la depuración y algún ajuste necesario para solucionar ciertos errores detectados en el desarrollo de todo el conjunto de las pruebas.

7.6.1 Control de cada uno de los drones

En esta prueba se evalúa que se pueda controlar de manera fiable cada uno de los drones configurados en la aplicación. Para poder realizar esta comprobación se realizaron vuelos no simultáneos con el Parrot Ar Drone 2.0 y el Parrot Bebop 2. En estos vuelos se tenían que verificar los siguientes requisitos:

- El control total del dron: el dron debe responder a todos los comandos enviados desde la aplicación
- Recepción de la telemetría: se debe comprobar que en el interfaz se muestra constantemente los datos actualizados del estado de la batería y la altura a la que se encuentra el dispositivo. Además el botón que sirve para enviar el comando de despegar y aterrizar tiene que cambiar en función si el dron está volando o en tierra.
- Reproducción del video. se debe poder visualizar el video que está transmitiendo el dron por streaming.

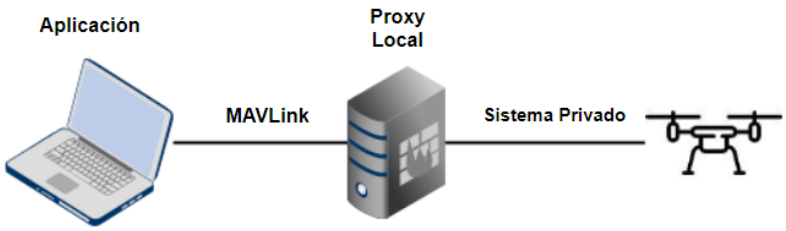

Figura 32. Diagrama de la prueba 1

El escenario de pruebas está compuesto por los siguientes elementos:

- La aplicación que se ejecuta en el portátil seleccionado anteriormente.
- El proxy local que tiene que ejecutarse en el mismo portátil donde se ejecuta la aplicación.
- Los dos drones seleccionados, que realizarán vuelos no simultáneos.

Resultados:

Parrot Ar Drone 2.0

	1	2	3	4	5
Control					X
Telemetría					X
Video				X	

Tabla 4. Resultados de la prueba 1 en el Ar Drone 2.0

Parrot Bebop 2

	1	2	3	4	5
Control					X
Telemetría					X
Video				X	

Tabla 5. Resultados de la prueba 1 en el Bebop 2

Durante las pruebas el funcionamiento fue todo perfecto menos por la reproducción del video donde se apreciaba algo de lag. Esto puede ser debido a que en ninguna de las dos librerías utilizadas para el control de los drones se permitía el envío directo de video desde el dron, por lo que se tuvo que realizar un reenvío adicional. Como no se pudo comprobar si el origen del lag era ocasionado por ello, no se puede afirmar el perfecto funcionamiento de la aplicación en este apartado.

Otra defecto que se notó durante las pruebas es que el Parrot Ar Drone 2.0 se ladeaba ligeramente hacia la izquierda y le costaba algo arrancar hacia delante. Se comprobó que con su propia aplicación de móvil el problema seguía ocurriendo, por lo que el fallo estaba en el propio aparato y no en la aplicación.

7.6.2 Control simultaneo de más de un dron

En esta prueba se debe comprobar que se puede manejar más de un dron a la vez y que no existe ninguna interacción entre ellos. Para realizar esta evaluación se realizó un vuelo simultáneo en el que se tenía que verificar los siguientes comportamientos:

- Los datos de telemetría de un dron no se intercambian con los del otro.
- Los comandos mandados a un dron son ejecutados solo por ese dron.
- Si se mandan comandos diferentes a los drones de manera simultánea, cada dron ejecutara el comando correcto.

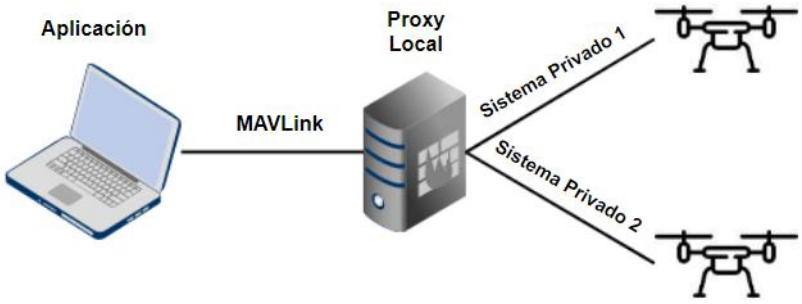

Figura 33. Diagrama de la prueba 2

El escenario de pruebas está compuesto por los siguientes elementos:

- La aplicación que se ejecuta en el portátil seleccionado anteriormente.
- El proxy local que tiene que ejecutarse en el mismo portátil donde se ejecuta la aplicación.
- Los dos drones seleccionados, que realizarán un vuelo simultáneo.

Resultados:

Parrot Ar Drone 2.0

	1	2	3	4	5
Telemetría					X
Control Independiente					X
Comando Simultaneo					X

Tabla 6. Resultados de la prueba 2 en el Ar Drone 2.0

Parrot Bebop 2

	1	2	3	4	5
Telemetría					X
Control Independiente					X
Comando Simultaneo					X

Tabla 7. Resultados de la prueba 2 en el Bebop 2

7.6.3 Implementación de un proxy externo

Para realizar esta prueba se requiere implementar un segundo proxy en el mini-ordenador Raspberry Pi 3 modelo B. Como la aplicación se ha programado en Java, aunque el sistema operativo que se ejecuta en el Raspberry es una distribución de Linux, no hace falta realizar ningún cambio adicional en el código para que funcione, solo es necesario cambiar las direcciones IP con las que se realiza la comunicación.

La comprobación del cumplimiento de esta prueba se realizó mediante un vuelo simultáneo donde cada dron se manejaba con un proxy diferente. En este vuelo se tiene que verificar:

- El control del dron, la recepción de la telemetría y la reproducción del video, deben tener el mismo funcionamiento con independencia del proxy a través del cual se haga la comunicación.

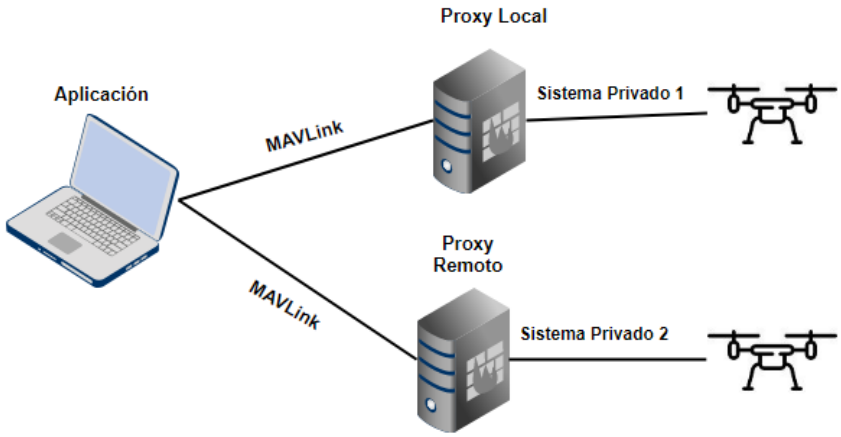

Figura 34.Diagrama de la prueba 3

El escenario de pruebas está compuesto por los siguientes elementos:

- La aplicación que se ejecuta en el portátil seleccionado anteriormente.
- El proxy local que tiene que ejecutarse en el mismo portátil donde se ejecuta la aplicación.
- El proxy remoto instalado en la Raspberry Pi
- Los dos drones seleccionados, que realizarán un vuelo simultáneo, pero realizando la comunicación a través de diferentes proxys.

Resultados:

Proxy Local

	1	2	3	4	5
Control					X
Telemetría					X
Video				X	

Tabla 8. Resultados de la prueba 3 desde el proxy local

Proxy Externo

	1	2	3	4	5
Control					X
Telemetría					X
Video				X	

Tabla 9. Resultados de la prueba 3 desde el proxy externo

En el desarrollo de esta prueba no se tuvo en cuenta el lag derivado de añadir un proxy externo debido a la proximidad en la que se encontraba.

Capítulo 8

Planificación y entorno socio-económico

8.1 Introducción

En este capítulo se van a describir tres cosas: al principio se explicara la planificación y las fases de desarrollo de las que ha constado el proyecto; después se presenta de un presupuesto donde se recogen los costes del mismo; y para finalizar, se detalla el posible impacto socio-económico que puede causar la solución final obtenida en el desarrollo del proyecto.

8.2 Planificación

8.2.1 Metodología

Para el desarrollo de este proyecto se ha seguido el modelo en cascada. Este modelo es un proceso secuencial, en el que el desarrollo del software depende de un conjunto etapas ejecutándose una tras otra, de manera que se tiene que haber terminado la primera para poder continuar con la segunda.

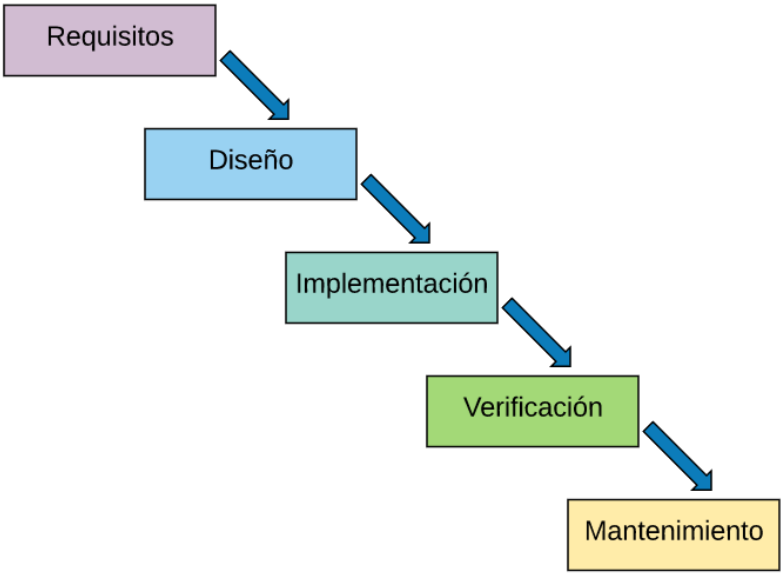

Figura 35. Metodología en cascada (Fuente: openclassrooms.com)

En la figura 35 se observan las etapas que contiene el modelo básico, en las cuales hay que realizar lo descrito a continuación:

- Requisitos: en esta etapa se definen todos los requisitos que debe cumplir el proyecto sin entrar en detalles técnicos. En esta fase hay que tener mucho cuidado porque con el proyecto avanzado no se podrá agregar ningún requisito más y hay que definir muy bien los objetivos que se debe cumplir.
- Diseño: esta es la etapa en la que se describe la estructura interna del software para conseguir cumplir los requisitos definidos en la etapa anterior.
- Implementación: Se programa un software que siga el diseño ya planteado para cumplir con todos los requisitos.
- Verificación: con el software ya terminado se plantean las pruebas que tiene que pasar para comprobar que no existan fallos.
- Mantenimiento: durante la verificación se realizan un conjunto de pruebas que permiten comprobar el correcto funcionamiento del producto. Pero es imposible realizar comprobar todas las circunstancias que se pueden presentar

a lo largo de la vida útil del software, por lo que existe el riesgo de que aparezcan fallos y habrá que repararlos. Además, en esta etapa también se incluye la necesidad de que haya que modificar el software, ya sea porque el entorno donde se ejecute cambie o por que se quiera añadir más funcionalidades.

Como todos los métodos, tiene sus ventajas y desventajas que lo hacen más útil en un tipo de proyectos que en otros.

Ventajas:

- Al realizar un planteamiento inicial en el que se define todos los requisitos que tiene que tener la solución final, no se producen incrementos de costes, tiempo y trabajo al tener que modificar el proyecto en medio del proceso.
- Es un modelo muy estructurado fácil de comprender e implementar

Desventajas:

- Existen pocos casos en los que se pueda definir todos los requisitos del proyecto desde el principio, sobre todo en el mundo comercial donde los requisitos los plantea un cliente externo.
- Es muy difícil aplicarlo a proyectos grandes donde puede ser necesario su modificación durante el proceso de ejecución.
- No se puede presentar al cliente un producto intermedio y hay que esperar a completarlo entero.

Pero en este caso en concreto, las desventajas quedan minimizadas ajustándose muy bien al tipo de proyecto que se ha desarrollado, debido a que se realiza por una sola persona, los objetivos están definidos desde el principio y no es grande.

8.2.2 Fases de desarrollo

A lo largo del desarrollo de este proyecto se han seguido las siguientes fases, las cuales se han ido ejecutando secuencialmente.

- Planteamiento: en esta primera fase realizó la propuesta del proyecto y los objetivos que queríamos cumplir.
- Estudio del sector: en esta fase se analizó el estado de arte, para conocer la evolución del sector y poder ajustar la solución final mejor en el mercado.
- Análisis del protocolo MAVLink: durante esta fase se ha realizado u estudio y análisis del protocolo MAVLink y entender su funcionamiento.
- Diseño: en esta fase se han tomado las decisiones para crear un diseño que luego pueda ser programado.
-

- Implementación: se ha realizado la programación de la aplicación partiendo de las decisiones hechas en la fase anterior.
- Validación: se han realizado una batería de pruebas que nos han permitido comprobar el funcionamiento correcto de lo proyectado y que cumple con todos los requisitos.
- Creación de la memoria: se ha desarrollado una memoria del proyecto donde se recopila toda la información de las fases anteriores.

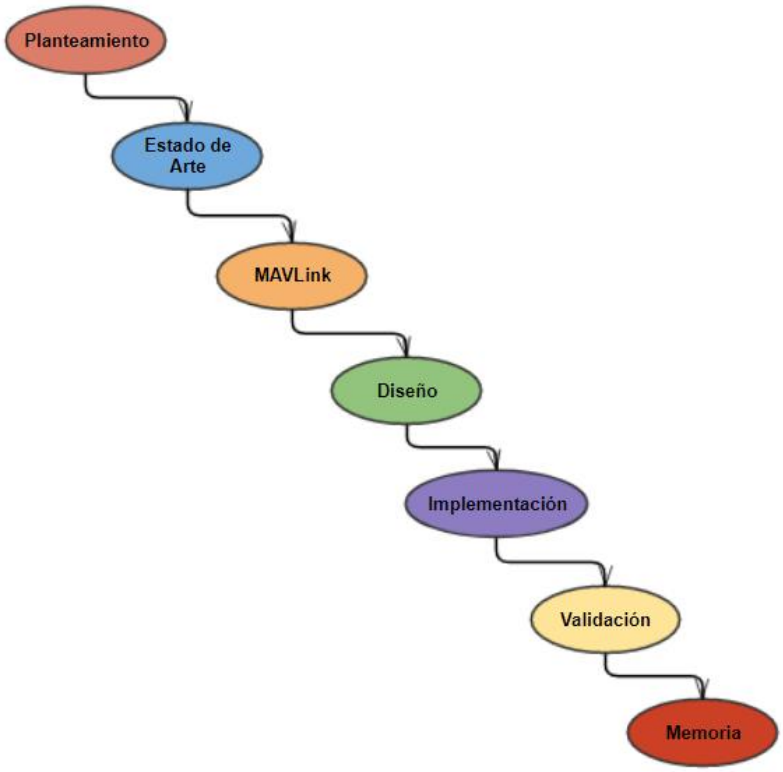

Figura 36. Fases del proyecto

8.2.3 Diagrama de Gantt

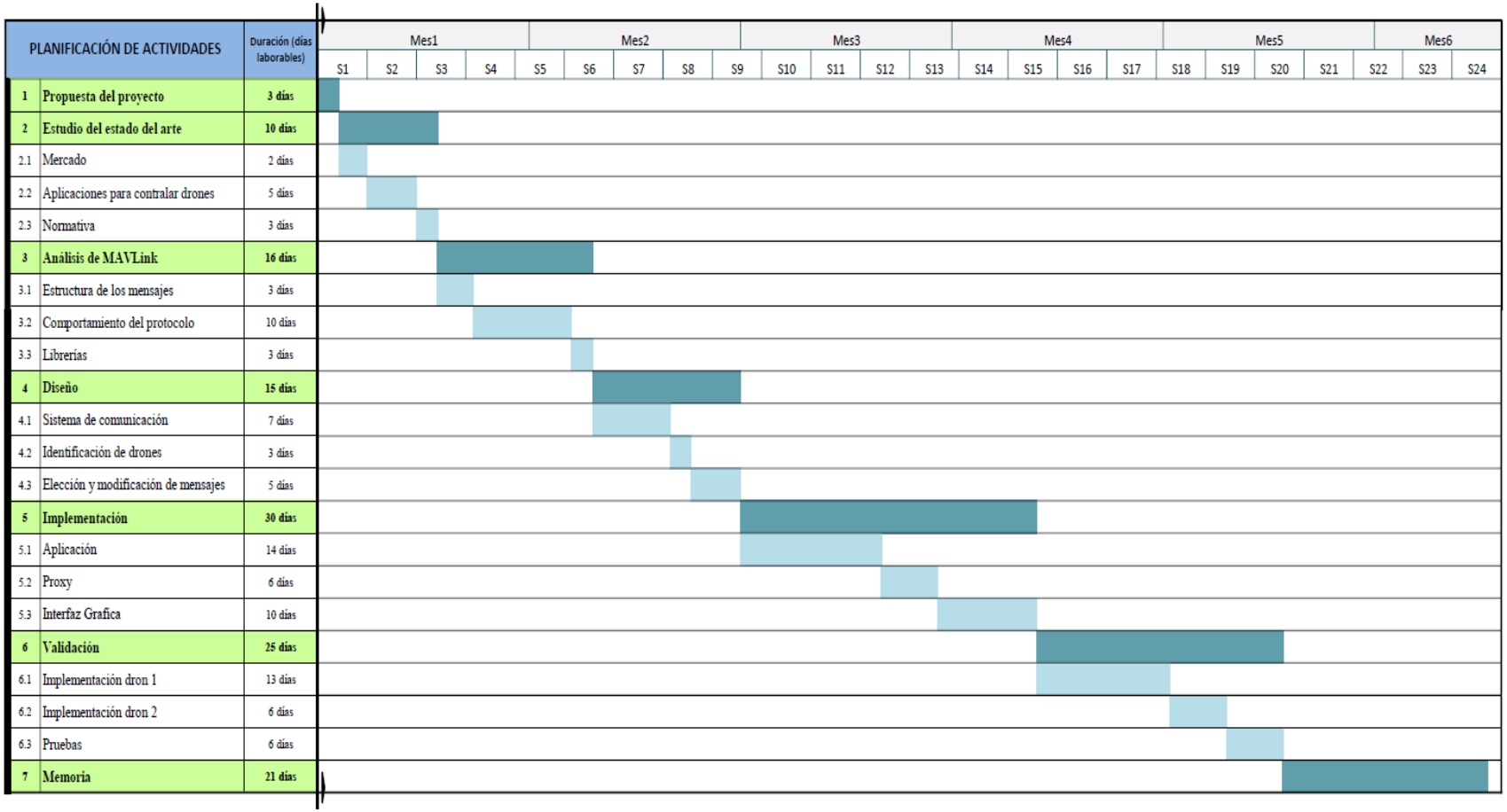

Figura 37. Diagrama de Gantt

8.3 Presupuesto

8.3.1 Costes Materiales

Para poder desarrollar la solución final y realizar las pruebas necesarias para la verificación de su funcionamiento, se han necesitado el uso de los dispositivos presentados en la tabla 10 y el software de la tabla 11.

Hardware	Coste (€)
Portátil Lenovo G500s	800,00
Raspberry Pi 3 modelo B	69,90
Parrot Ar Drone 2.0	210,00
Parrot Bebop 2	480,00
Total	**1559,90**

Tabla 10. Costes hardware

Software	Coste (€)
Microsoft Windows 10*	0,00
Raspbian	0,00
NetBeans IDE 8.2	0,00
Microsoft Office Profesional 2016	480,00
Total	**480,00**

Tabla 11. Costes software

(*) Incluido en el precio del portátil Lenovo.

Para poder evaluar los costes reales, se debe tener en cuenta la depreciación de los materiales usados a lo largo de su vida y su uso proporcional en los cinco meses y medio que se han necesitado para la realización del proyecto.

En el mercado actual los componentes físicos se devalúan muy rápidamente en poco tiempo. Contando con esto, y estimando que la vida útil de los dispositivos es de 4 años, se ha considerado que el gasto real incurrido es un 15% del valor original.

Al contrario que pasa con el hardware, el software usado cuenta con una vida útil larga y su precio de mercado se mantiene por más tiempo, debido a esto solo se va a considerar el 10% de su coste original.

	Coste (€)	Coste Efectivo (€)
Hardware	1559,90	233,99
Software	480,00	48,00
Total	**2039,00**	**281,99**

Tabla 12. Costes materiales reales

8.3.2 Costes Personales

El tiempo necesitado para la conclusión final del proyecto ha sido de cinco meses y medio, esto ha supuesto trabajar 120 días en jornada completa de 8 horas.

Este proyecto ha sido dirigido por el Dr. Francisco Valera Pintor, que ha invertido en su desarrollo aproximadamente unas 80 horas de trabajo.

Dado que es difícil determinar el valor del coste/ hora de una persona cuando no media un salario o contrato, para basarse en algo real y que sirva como referencia al menos aproximada de este valor, se han considerado los datos públicos que aparecen en la *"Resolución del Director General de asuntos económicos del Ministerio de Defensa……."* [Res] por la que se aprueban las tarifas que la empresa de ingeniería ISDEFE adscrita al Ministerio de Defensa, tiene con dicho Ministerio.

- Ingeniero Junior: tarifa facturable por ISDEFE al M. de Defensa: 32,32 €.
- Jefe de Proyecto: tarifa facturable por ISDEFE al M. de Defensa: 56,40 €.

Según dicha resolución estas tarifas incluyen costes diversos y margen por valor de un 8%, por lo que la tarifa de coste sería:

- Ingeniero Junior: coste/hora: 29,93 €.
- Jefe de Proyecto: coste/hora: 52,22 €.

	Horas	Coste/Hora (€)	Coste Total (€)
Ingeniero junior	960	29,93	28.732,80
Director del proyecto	80	52,22	4.177,60
Total			**32.910,40**

Tabla 13. Costes personales

8.3.3 Costes totales

El coste total del proyecto se deriva de la suma de costes materiales y personales. En los precios que hemos visto en los apartados anteriores ya estaba incluido el impuesto del IVA, por lo que no es necesario agregarlo de nuevo.

	Costes (€)
Material	281,99
Personal	32.910,40
Total	**33.192,39**

Tabla 14. Constes totales

El presupuesto total de este proyecto asciende a la cantidad de TREINTA Y TRES MIL CIENTO NOVENTAY DOS EUROS CON TREINTA Y NUEVE CÉNTIMOS.

8.4 Impacto socio-económico

Con la evolución actual del mercado de los drones donde cada vez se fabrican más dispositivos y se está reduciendo el precio de los mismos, lo que permite que el público general pueda disponer de uno o más drones.

El principal motivo por el que se ha realizado este proyecto es solucionar el problema que se puede encontrar un usuario final al querer manejar diferentes modelos de drones. La aplicación desarrollada permite controlar de la misma forma todos los drones que están configurados en la misma, por lo que el usuario no tendría que depender de más de una aplicación para controlar sus drones y aprender el control de cada uno por separado.

Solo con la funcionalidad que se acaba de describir se podría decir que la integración de la aplicación en el mercado puede tener un impacto positivo, pero la solución final cuenta con otras dos características que la hacen muy atractiva, manejar más de un dron a la vez y poder comunicarnos con los drones a través de diferentes proxys intermedios.

La posibilidad de manejar más de un dron a la vez puede permitir que se reduzca el tiempo de inspección de una obra o disponer de varias cámaras móviles para grabar eventos, pero estos son solo dos de los ejemplos de la infinidad que el mercado puede dar a esta función. Es verdad que hay que tener en cuenta lo explicado en el apartado 3.5, donde se hace referencia al marco regulador actual, donde se comenta que aunque actualmente no hay restricciones en cuanto al número de drones que podemos volar a la vez, existe un borrador de una ley futura donde este número se limita a uno. Pero estas leyes solo se aplican al espacio abierto de España, por lo que si la aplicación se utiliza en otro país o dentro de un recinto cerrado, no habría ningún problema.

La función principal que se le puede dar a la utilización de un proxy externo en la comunicación con los drones, es ampliar la distancia desde la que es posible controlar éstos y así no necesitar del uso de aparatos que emitan señales muy potentes que puedan ocasionar problemas en el espacio radioeléctrico. Pero si el proxy lo instalamos en un aparato tan pequeño como una Raspberry Pi y lo colocamos encima de un dron, conseguimos un repetidor móvil para otros drones. Esto puede ser muy útil para salvamento civil o en la extinción de grandes incendios donde la cobertura de un dron puede no llegar a abarcar todo el territorio del mismo.

Como se observa, los usos que el mercado y la sociedad pueden dar a la aplicación son grandes, por lo que se considera que el impacto socio-económico obtenido con su desarrollo es positivo.

Capítulo 9

Conclusiones y trabajos fututos

9.1 Introducción

En este capítulo se comentan las conclusiones obtenidas durante el desarrollo del proyecto y se exponen algunas funcionalidades que se pueden añadir en un futuro a la solución final para hacerla más completa.

9.2 Conclusiones

9.2.1 Resultados obtenidos

No hay nada más frustrante que trabajar durante mucho tiempo en un proyecto para que, cuando se termine, la solución final obtenida no sea satisfactoria. Este no ha sido el caso, ya que la aplicación que se ha desarrollado cumple con todos los objetivos que nos habíamos fijado desde el inicio del proyecto. El resultado ha sido un programa que tiene las siguientes funciones:

- Puede controlar diferentes modelos de drones con una interfaz gráfica común para todos ellos. Lo que permite que el usuario solo tenga que aprender un solo modo de control.
- Se puede agregar cualquier dron del mercado, siempre que el protocolo usado para comunicarnos con él sea público y se pueda programar. Esta tarea no está pensada para que la realice el usuario final.
- Si hay más de un dron configurado en la aplicación, se pueden realizar vuelos simultáneos.
- Se pueden agregar uno o más proxys externos en la comunicación con los drones que nos pueden permitir por ejemplo ampliar la cobertura de control.

9.2.2 Opinión personal

Además de la satisfacción por haber obtenido un resultado final que cumple con todos los objetivos iniciales y del conocimiento obtenido en el trascurso de su desarrollo, este proyecto me ha servido para darme cuenta de lo complicado que es trabajar con nuevas tecnologías. Durante la realización del proyecto se ha tenido que analizar y utilizar el protocolo MAVLink, este protocolo no hace mucho tiempo que se creó y no existe mucha información sobre él, por lo que su compresión no ha sido sencilla.

Otra de las conclusiones obtenidas durante la creación de la aplicación ha sido la confirmación de la necesidad de la misma. Para poder realizar las pruebas además del protocolo MAVLink, se ha tenido que aprender los protocolos usados por los drones usados en dichas pruebas. Esto ha requerido de una gran cantidad de tiempo y esfuerzo que la aplicación puede evitar a un usuario final.

9.3 Trabajos futuros

9.3.1 Agregar funciones de piloto automático

MAVLink es un protocolo pensado para ser utilizado en sistemas de piloto automático, por lo que dispone de muchos mensajes definidos para ese objetivo. Agregar esta función podría constar de dos fases:

1. Ampliar la parte de la estación de control para que se puedan crear planes de vuelo. En esta fase solo se podrían enviar los planes a drones que dispongan

ya dispongan de un sistema de piloto automático instalado que sea compatible con MAVLink.

2. Instalar en el mini-ordenador Raspberry Pi un sistema de piloto automático y colocarlo encima del dron. De esta manera se podría conseguir que todos los drones dispongan de esta función.

9.3.2 Control único

En la aplicación actual cada dron se tiene que controlar mediante un panel de control independiente. Si agregamos un panel común desde el cual se puedan dar órdenes a más de un dron a la vez, se podría conseguir una función muy atractiva para, por ejemplo, demostraciones aéreas.

9.3.3 Mejorar la seguridad

El protocolo MAVLink está diseñado para que sea lo más rápido y ligero posible, pero este objetivo seguramente ha tenido la consecuencia de que no disponga de más seguridad que una simple comprobación de errores CRC.

Aunque no podemos modificar el protocolo que se usa para comunicarse directamente con los drones, ya que estos no lo entenderían, si se puede modificar el protocolo usado entre la aplicación y el proxy intermedio. De esta manera conseguiríamos otorgar de más seguridad a la aplicación.

9.3.4 Crear una aplicación para teléfonos móviles

Los teléfonos móviles actuales cada vez disponen de más potencia y memoria y esto está permitiendo crear aplicaciones para ellos cada vez más complejas y pesadas. De esta manera sería factible trasformar el programa actual que necesita del uso de un ordenador, en una aplicación para teléfonos móviles que son más fáciles de usar y transportar.

Referencias

[Ori] Origen y desarrollo de los drones:

http://drones.uv.es/origen-y-desarrollo-de-los-drones/

[His] Historia de los drones:

http://eldrone.es/historia-de-los-drones/

[TFD] The flight of "drone" from bees to planes:

https://fortunascorner.com/2013/07/28/the-flight-of-drone-from-bees-to-planes/

[DDD] Department of Defense Dictionary of Military and Associated Terms:

http://www.bits.de/NRANEU/others/jp-doctrine/jp1_02(00).pdf

[Uso] Los 14 usos de drones que seguro no conocías:

http://agencia.donweb.com/los-14-usos-de-drones-que-seguro-no-conocias/

[Lif] El sistema que encuentra a los desaparecidos gracias al móvil:

http://www.elmundo.es/economia/2014/06/17/539f2455268e3e231c8b456b.html

[Cho] Choosing a Ground Station:

http://ardupilot.org/copter/docs/common-choosing-a-ground-station.html

[Ard] ArduPilot:

http://ardupilot.org/ardupilot/index.html

[Px4] Px4 Pro:

http://px4.io/

[ArM] ArduPilot Mega:

http://www.ardupilot.co.uk/

[Pix] Pixhawk:

https://pixhawk.org/

[Mav] MAVLink:

http://qgroundcontrol.org/mavlink/start

[Pro] MAVLink protocolo de comunicación para drones:

http://www.xdrones.es/mavlink/

[CoM] Codigo MAVLink:

https://github.com/mavlink/mavlink/

[Ley14] Ley 18/2014:

https://www.boe.es/buscar/doc.php?id=BOE-A-2014-10517

[Rea] Borrador Real Decreto sobre la nueva legislación para drones:

https://www.fomento.gob.es/NR/rdonlyres/63ECAE3A-B29E-45A7-A885-D314153883EE/139826/RDRPAS27102016.pdf

[Ley60] Ley 48/1960:

https://www.boe.es/buscar/doc.php?id=BOE-A-1960-10905

[Rea14] Real Decreto 552/2014:

https://www.boe.es/buscar/doc.php?id=BOE-A-2014-685

[Dro] Drones España:

https://solodronesbaratos.com/drones-espana/

[Rec] Uso recreativo de drones dudas:

http://asegurandolo.es/uso-recreativo-de-drones-dudas/

[Fol] Folleto hobby drones:

*http://www.seguridadaerea.gob.es/media/4629078/folleto_*

hobby_drones.pdf

[Not] NOTAM:

https://es.wikipedia.org/wiki/NOTAM

[DrRe] Drones Recomendados:

https://solodronesbaratos.com/drones-recomendados/

[Ras] Raspberry Pi 3 Model B:

https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[PrA] Protocolo Ar Drone 2.0:

https://github.com/felixge/node-ar-drone

[PrB] Protocolo Bebop2:

https://github.com/hybridgroup/node-bebop

[Mod] Modelo en cascada:

https://openclassrooms.com/courses/gestiona-tu-proyecto-de-desarrollo/en-que-consiste-el-modelo-en-cascada

[Res] *Resolución del Director General*:

https://www.isdefe.es/ckfinder/userfiles/files/Tarifas%20ISDEFE%202016.pdf